Int. J. Inf. Secur.
DOI 10.1007/s10207-015-0312-y

REGULAR CONTRIBUTION

ASICS: authenticated key exchange security incorporating certification systems

Colin Boyd[1] · **Cas Cremers**[2] · **Michèle Feltz**[3] · **Kenneth G. Paterson**[4] · **Bertram Poettering**[5] · **Douglas Stebila**[6]

Abstract Most security models for authenticated key exchange (AKE) do not explicitly model the associated *certification system*, which includes the certification authority and its behaviour. However, there are several well-known and realistic attacks on AKE protocols which exploit various forms of malicious key registration and which therefore lie outside the scope of these models. We provide the first systematic analysis of *AKE security incorporating certification systems*. We define a family of security models that, in addition to allowing different sets of standard AKE adversary queries, also permit the adversary to register arbitrary bitstrings as keys. For this model family, we prove generic results that enable the design and verification of protocols that achieve security even if some keys have been produced maliciously. Our approach is applicable to a wide range of models and protocols; as a concrete illustration of its power, we apply it to the CMQV protocol in the natural strengthening of the eCK model to the ASICS setting.

Keywords Authenticated key exchange (AKE) · Unknown key share (UKS) attacks · Certification authority (CA) · Invalid public keys · PKI

Mathematics Subject Classification 94A60

✉ Bertram Poettering
bertram.poettering@rhul.ac.uk; bertram.poettering@rub.de

Colin Boyd
colin.boyd@item.ntnu.no

Cas Cremers
cas.cremers@cs.ox.ac.uk

Kenneth G. Paterson
kenny.paterson@rhul.ac.uk

Douglas Stebila
stebila@qut.edu.au

1 Norwegian University of Science and Technology, Trondheim, Norway

2 University of Oxford, Oxford, UK

3 National Commission for Data Protection, Esch-sur-Alzette, Luxembourg

4 Royal Holloway, University of London, Egham, UK

5 Ruhr University Bochum, Bochum, Germany

6 Queensland University of Technology, Brisbane, Australia

1 Introduction

After public key encryption and digital signatures, authenticated key establishment (AKE) is perhaps the most important public key primitive. From a real-world perspective, AKE protocols relying on public key techniques are widely deployed in systems that are used every day by billions of users, including systems such as TLS, IPsec, SSH, and various single sign-on systems. From a theoretical perspective, formal, cryptographically sound modelling for AKE protocols began in the symmetric setting with the seminal work of Bellare and Rogaway [5], and was later extended to the public key setting [7]. Since then, there has been a large body of work in this tradition, and many additions and modifications have been proposed. The most prominent current models in this tradition [4,14,32,43] strengthen or add to the required security properties, cover different protocol classes, and strengthen adversary powers.

Despite intensive study over two decades, important elements of AKE protocols have not been sufficiently modelled, preventing our deeper understanding of this important primitive and limiting its applicability to real-world protocols. Specifically, the public key infrastructure (PKI) needed to support the authenticity of public keys in AKE, and the

Published online: 04 January 2016

interactions between the certification authority (CA), honest parties, and the adversary, are rarely modelled. Rather, with exceptions as noted below, in typical AKE models and proofs it is assumed that all public keys are honestly generated and authentically distributed at the start of the security game, and that there is a single key per party; certificates are excluded from the model. The adversary can corrupt parties, learning all their secrets, but has limited ability to register malicious keys. Roughly speaking, this modelling approach corresponds to an ideal CA, who zealously generates perfect key pairs and securely distributes them to the correct parties.

However, CAs in the real world simply do not operate in such rigorous ways. They have differing strengths of procedures for checking claimed identities,[1] so malicious parties might in some cases get arbitrary public keys certified against identifiers of their choice. The most egregious examples involve CAs who, either willingly, under coercion, or as a result of security compromises, have issued certificates for keys and identifiers that they should not have.[2] CAs following best practices may require that a user requesting a certificate submit a certificate signing request (CSR, as per [44]) to the CA. This involves the user self-signing the data that is to be certified. Various standards [1,2,41] include other approaches to providing *proofs of possession*. However, even these basic tests of private key ownership are not mentioned in industry guidelines issued by the CA/Browser Forum [12,13]. Furthermore, these procedures all fall short of the proofs of knowledge [40] required to match what is assumed in typical AKE models. Thus, an attacker may be able to register another party's public key under his own identifier or register a malformed key which then interacts with properly generated keys in an unfortunate way.

Critically, there are realistic attacks on AKE protocols which cannot be captured by AKE security models that omit CA and PKI aspects:

- Kaliski's *unknown key share* (UKS) attack [28] on early versions of MQV exploits the ability of the adversary to dynamically register a public key (which is valid and for which the adversary *does* know the secret key).
- The UKS attack on KEA described by Lauter and Mityagin [33, p. 380] exploits the adversary's ability to re-register some party's static public key as his own public key.
- Blake-Wilson and Menezes [9] introduced the *duplicate-signature key selection* (DSKS) attack on signature schemes: after observing a user's signature σ on a message m, the adversary E is able to compute a signature key pair $(\mathsf{sk}_E, \mathsf{vk}_E)$ (or sometimes just a verification key vk_E) such that σ is also E's signature on the message m. Now, for example, if the Station-to-Station (STS) protocol is implemented using a signature scheme that is vulnerable to DSKS attacks, and the adversary can register arbitrary public keys with the CA, then the protocol is vulnerable to an online UKS attack [9].
- In Lim and Lee small subgroup attacks [34], the adversary extracts partial (or even complete) information about a party's long-term secret key. Some of these attacks require registering invalid public keys with the CA before engaging in protocol runs with honest participants. Of particular note are the Lim–Lee-style attacks of Menezes and Ustaoglu [37] on the HMQV protocol [30].

We claim that to date there has been no *systematic* treatment in the literature of the behaviour of CAs with respect to public keys and identifiers chosen by the adversary and, relatedly, the treatment of public keys and ephemeral values by protocol participants (especially with respect to group membership testing). Our paper sets out to rectify this situation, providing a comprehensive and self-contained treatment of these features, as well as establishing generic results to make protocols resilient against such attacks.

1.1 Contributions

Our paper has three main contributions.

First, we present in Sect. 2 a framework for reasoning about the security of AKE protocols with respect to various CA key registration procedures. This framework allows us to capture several attacks based on adversarial key registration, including UKS attacks, small subgroup attacks, attacks that occur when the CA does not check if public keys are registered twice, and attacks that occur when multiple public keys can be registered per identifier.

Second, we provide in Sect. 3 a generic approach to achieve strong security guarantees against adversaries that can register arbitrary public keys for certain types of protocols. In particular, we show how to transform Diffie–Hellman-type AKE protocols that are secure in a model where only honest key registration is allowed into protocols that are secure even when adversaries can register arbitrary valid or invalid public keys. In such cases, security is still guaranteed for all sessions (that were considered *clean* or *fresh* in the base model) except those in which the peer's public key is valid but registered by the adversary.

Third, we demonstrate in Sect. 4 how our methodology can be used to establish strong security guarantees, even when

[1] For example, issuance of Extended Validation (EV) certificates requires stronger identity-checking requirements than non-EV certificates, see https://www.cabforum.org/certificates.html for more details.

[2] In June and July 2011, Dutch CA DigiNotar was hacked [23], with the intruder taking control of all 8 of the CA's signing servers; at least 531 rogue certificates were then issued. In August 2011, TURKTRUST CA [21] issued special certificates with wildcard signing capabilities, allowing impersonation of any domain in the Internet. This was discovered, by coincidence, only 18 months later.

the adversary can register arbitrary public keys, for concrete protocols such as CMQV, NAXOS, and UP, using CMQV as a running example. We provide in Sect. 5 recommendations for the design of protocols that are secure in our models.

This article extends our previous conference paper [11].

1.2 Related work

The original computational model for key exchange of Bellare and Rogaway [5] has a long-lived key generator, which is used to initialize all parties' keys at the start of the game. This is a standard part of most computational models today. However, in common with several later models [14,27,31], the adversary cannot influence long-term keys: only honestly generated keys are considered. Starting with the 1995 model of Bellare and Rogaway [6], it was recognized that the adversary may be able to choose long-term keys for certain parties, whether public keys or symmetric keys. It is possible to identify three different methods that have been used to model such an adversary capability.

1. The adversary can replace long-term keys by providing them as an input to a corrupt query. This was the method used originally by Bellare and Rogaway [6] and was subsequently used in the public key setting by others [8,38].
2. The adversary is allowed to generate arbitrary keys for corrupted parties at any time during the protocol run [30].
3. An additional query is added specifically to set up a user with a new key chosen by the adversary [16,25,46]. This query is typically called establishparty and takes as input the user name and its long-term public key.

These methods allow the models to capture the Kaliski attack [28], which requires the adversary to register a new public key after certain protocol messages have been obtained. However, none of these currently used methods has the generality of our model and, in particular, all of them omit the following realistic features:

– registration of multiple public keys per user;
– flexible checking by certification authorities via a verification procedure;
– adversarial choice of public keys per session.

Special mention should also be made of the model of Shoup [43]. Unlike most popular AKE models today, it uses a simulatability definition of security comparing ideal and real-world views. Security is defined to mean that for any real-world adversary there is an ideal world adversary (benign by definition) such that the transcripts of the two are computationally indistinguishable. Real-world adversaries have the ability to assign users to public key certificates. Shoup's model has not been widely used and the examples in [43] are not fully worked through. Furthermore, the model cannot represent an adversary who obtains only ephemeral secret keys without knowing the long-term key of the same user and therefore cannot capture security properties common in more modern models.[3]

Cash et al. [15] and Hofheinz et al. [24] have considered the security of non-interactive key exchange (NIKE) in settings where the adversary can register arbitrary public keys, analogously to our ASICS setting for interactive key exchange. It is an interesting open problem to examine how the security models and constructions for NIKE [15,24] can be built upon to achieve security in the ASICS setting.

Boldyreva et al. [10] model certification and PKI for public key encryption and signatures. They point to legitimate concerns regarding what certification authorities can be expected to check in practice. It is interesting to note that while they use key exchange as a prime motivating example for modelling certification, key exchange is not explored in their models. Nevertheless, this and related follow-up work [10,22] are in the spirit of our work: like ours, their models allow (amongst other things) corrupt key registration. However, they do not allow the adversary to reveal long-term or ephemeral keys, which is fundamental to modern AKE security models.

Critically, all of the approaches mentioned above have only been used to establish results for a handful of specific protocols. In contrast, we establish *generic results* that facilitate the design and verification of AKE protocols and that can be applied to a large class of protocols.

2 ASICS model family

In this section, we define a parameterized AKE security model that allows for explicit modelling of the certification of public keys. Prominent AKE security frameworks can be instantiated in this family of models, as well as extensions that allow dynamic adversarial registration of arbitrary bitstrings as public keys.

Generally speaking, from a user's point of view, participation in key exchange encompasses three consecutive phases: First, users set up their individual key pairs; more precisely, each user invokes a randomized algorithm KeyGen that outputs a fresh secret key/public key pair (sk, pk). Second, users contact a certification authority (CA) to get their keys certified: each user provides the CA with its identifier $\hat{P}$ and its public key pk, and obtains a certificate C that binds the identifier to the key. After completing these setup steps, in the third phase, users can engage in interactive sessions with other users to establish shared keys.

[3] An early version of [43] from April 1999 in fact did allow revealing session state without simultaneously revealing long-term secrets. However, this option disappeared in all later versions of the paper.

To do so, they usually require knowledge of their own key pair $(\mathsf{sk}, \mathsf{pk})$, their identifier $\hat{P}$, and the corresponding certificate C. In addition to that, protocols may require a priori knowledge of (a subset of) the peer's public key pk', peer's identifier $\hat{Q}$, and peer's certificate C'. As we will see, our execution model is general enough to cover all these settings.

To ease notation, we assume that public key pk and identifier $\hat{P}$ can be readily derived from any certificate C; we use notation $C.\mathsf{pk} = \mathsf{pk}$ and $C.\mathsf{id} = \hat{P}$ correspondingly. This assumption holds for all practical PKIs we are aware of, and in particular for X.509 certificates (as used in SSL/TLS); the latter carry public key and identifier in a canonically formatted data structure. In potential other cases, where keys and identifiers cannot be extracted from certificates but instead are auxiliary input to the verification routine, certificates can still be expected to be valid for only a single key/identifier pair, i.e., the mapping from C to compatible pk and $\hat{P}$ is again unambiguous. Proposed notations $C.\mathsf{pk}$ and $C.\mathsf{id}$ are hence meaningful also in such cases.

Here, we focus on the interaction between users and the CA that occurs in the certification process. We enable the modelling of different degrees of rigour in the checks of consistency and ownership of public keys pk presented to the CA. On the one hand, CAs could be pedantic with such verifications (e.g., require a proof of knowledge[4] of the secret key corresponding to pk); on the other hand, CAs could also just accept any given bitstring pk as valid and issue a certificate on it. The ability to precisely assess the security of key establishment in the face of different CA behaviours is a key contribution of our new model family.

Definition 1 An *ASICS protocol* Π consists of a set of domain parameters, a key generation algorithm KeyGen, a public key verification procedure VP, and the protocol description π that describes how key exchange protocol messages are generated and responded to as well as how the session key is derived.

We denote by VP the specific *verification procedure* on public keys and identifiers that a considered CA deploys. As different checks on pk and $\hat{P}$ might require different levels of interaction between the registering user and the CA, for example with the use of state by the CA, we model it as a procedure, as opposed to a function. VP takes as input a public key and an identifier. To enable the specification of realistic VP behaviour, we additionally allow the definition of VP to access elements of the game state, as we will see in Example 1. We require that VP is efficient and has binary output. Furthermore, we require that the CA issues the requested certificate only if VP outputs value 1; all certification requests where VP outputs value 0 are rejected. Note that, for simplicity, we only consider non-interactive verification procedures (i.e., two-message registration protocols) between the user and the CA. A more general treatment covering interactive verification procedures as well would introduce additional complexities to our framework.

Specific key exchange protocols might be insecure for one (liberal) instantiation of VP, and be secure for another (stricter) one. Note that CAs that do not perform any check on pk and $\hat{P}$ are modelled by a verification procedure VP that always outputs 1. A verification procedure that performs few checks may output 1 for at least all $\mathsf{pk} \in \mathbf{PK}$, where $\mathbf{PK}$ denotes the set of possible public keys output by KeyGen. Precisely, if the input of algorithm KeyGen is security parameter 1^k and the used randomness is $r \in_R \{0,1\}^k$, then we define

$$\mathbf{PK} = \big\{\mathsf{pk} \mid \exists r, \mathsf{sk} : r \in \{0,1\}^k \wedge \mathsf{KeyGen}(1^k; r) = (\mathsf{sk}, \mathsf{pk})\big\}.$$

A verification procedure with high assurance may require a zero-knowledge argument that the requester knows the secret key corresponding to the public key, and even that the key was generated verifiably at random. Note that we allow VP to keep an internal state between invocations; our model hence covers possible implementations of CAs that reject certification requests with public keys that have already been registered (e.g., for a different identifier).

2.1 Security model

At a high level, our model stipulates users that generate one or more keys, obtain certificates for these keys from a CA, and use keys and certificates to run (potentially concurrent) sessions of the key agreement protocol. Similar to other security models [5,14], the adversary controls all communication in these sessions, corrupts users at will to obtain their secret keys, and arbitrarily reveals established session keys. Innovative is the adversary's additional ability to steer the registration process with the CA: it can obtain from the CA valid certificates for public keys and identifiers of its choosing (as long as VP evaluates to 1), and can forward such certificates to users.

To keep our model simple and comprehensible, we abstract away any forgeability issues of certificates and assume the following ideal functionality: no certificate will be considered valid unless it has been issued by the CA. We

[4] Although interactive or non-interactive zero-knowledge proof systems seem to yield ideal solutions in this context, standards like X.509, OpenPGP, and PKCS#10 content themselves with the purely heuristic (and inferior) approach of demanding a so-called *proof-of-possession* (PoP), mostly implemented via self-signed certificates or self-signed certificate requests (cf. [1,41]). The security of such constructions seems to be difficult to formally assess [40].

Table 1 Elements of session state

acert	Certificate of the actor (the user running this session)
pcert	Certificate of this session's peer
role	Taken role; either $\mathcal{I}$ (initiator) or $\mathcal{R}$ (responder)
sent	Concatenation of all messages sent in this session
rcvd	Concatenation of all messages received in this session
status	Session status; either `active`, `accepted`, or `rejected`
key	Key in $\{0,1\}^k$ established in this session
rand	Randomness used in this session
data	Any additional protocol-specific data

model this by letting the challenger keep a list $\mathcal{C}$ of all CA-issued certificates and by equipping users with a *certificate verification oracle* $\mathcal{O}_{CV}$ that checks membership in that list; concretely, we assume that $\mathcal{O}_{CV}(C) = 1 \Leftrightarrow C \in \mathcal{C}$. Of course, in concrete implementations, this oracle is replaced by an explicit local verification routine; for instance, if certification is implemented via a signature scheme, this will include its verification procedure (besides further checking of validity, chain validity, revocation state, etc.).

2.1.1 Sessions and session state

Users, once they have created their keys and obtained corresponding certificates, can execute protocol sessions. Within a user, each such session is uniquely identified by a pair $s = (C, i)$, where C denotes the certificate used by the user (by himself) in that session, and i is a counter. The user maintains session-specific variables as indicated in Table 1. Some session variables are fixed upon session creation, whereas others can be assigned or updated during protocol execution. Some, such as pcert, status, and key, are considered to be outputs of the key establishment and might be used in higher-level protocols or applications. A session s has *accepted* if $s_{\text{status}} = \texttt{accepted}$.

2.1.2 Adversarial queries

The adversary interacts with users by issuing queries. The adversary can direct users to establish long-term key pairs and certificates (kgen, hregister), to initiate protocol sessions (create), and to respond to protocol messages (send). The adversary may be able to learn long-term keys (corrupt), session-specific randomness (randomness), or session keys (session-key) from users. The adversary can also maliciously obtain certificates from the CA (pkregister, npkregister).

The queries in set $\mathcal{Q}_N = \{\text{kgen}, \text{hregister}, \text{create}, \text{send}\}$, defined as follows, model normal operation of the protocol; they are required in any security model. Initially, the auxiliary variables $\mathcal{HK}, \mathcal{C}, \mathcal{C}_h, \mathcal{C}_{\mathsf{pk}}$, and $\mathcal{C}_{\mathsf{npk}}$ are set to $\emptyset$.

- kgen () By running algorithm KeyGen, a fresh key pair (sk, pk) is generated. Public key pk is returned to the adversary; secret key sk is stored for processing potential later queries corresponding to pk. The public key is added to the set of honestly generated keys: $\mathcal{HK} \leftarrow \mathcal{HK} \cup \{\mathsf{pk}\}$.
- hregister(pk, $\hat{P}$) The query requires that $\mathsf{pk} \in \mathcal{HK}$ and that VP outputs 1 on input pk[5] and $\hat{P}$; otherwise, it returns $\perp$. The public key pk is registered at the CA for the identifier $\hat{P}$. The resulting certificate C is added to the global set of certificates and to the set of honestly generated certificates: $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$ and $\mathcal{C}_h \leftarrow \mathcal{C}_h \cup \{C\}$. The query returns C.
- create $\left(s = (C, i), r, \left[C'\right]\right)$ The query requires that $C \in \mathcal{C}_h$, that a session with counter i for certificate C does not already exist, and that $r \in \{\mathcal{I}, \mathcal{R}\}$; otherwise, it returns $\perp$. A new session s is created for the user with public key C.pk and identifier C.id. Session variables are initialized as

$$(s_{\text{acert}}, s_{\text{pcert}}, s_{\text{role}}, s_{\text{sent}}, s_{\text{rcvd}}, s_{\text{status}}, s_{\text{key}}) \leftarrow (C, \perp, r, \epsilon, \epsilon, \texttt{active}, \perp).$$

  If the optional certificate C' is provided, we set $s_{\text{pcert}} \leftarrow C'$. In addition, a string in $\{0,1\}^k$ is sampled uniformly at random and assigned to s_{rand}; we assume that all randomness required during the execution of session s is deterministically derived from s_{rand}. The user also runs the initialization procedure for the key exchange protocol, which may further initialize its own (internal) state variable s_{data} and optionally generate a message M. If M was generated, set $s_{\text{sent}} \leftarrow M$, and return M; otherwise, return $\perp$.
- send (s, M) The query requires that session s exists and that $s_{\text{status}} = \texttt{active}$; otherwise, it returns $\perp$. The user continues the protocol execution for this session with incoming message M, which may optionally generate a response message M'. Next, s_{rcvd} is set to $(s_{\text{rcvd}} \parallel M)$ and, if M' is output, s_{sent} is set to $(s_{\text{sent}} \parallel M')$. The protocol execution may (re-)assign values to s_{status} and s_{key}, and to the session's internal state variable s_{data}. Also, if the value s_{pcert} was not provided to the create query, then protocol execution may assign a value to s_{pcert}. (For example, in TLS the client does not learn the server's certificate until it receives the `ServerCertificate` message.) If M' was generated, return M'; otherwise return $\perp$.

[5] Reasonable implementations of VP output 1 on all keys $\mathsf{pk} \in \mathcal{HK}$, because $\mathcal{HK} \subseteq \mathbf{PK}$.

Remark 1 (Multiple CAs) Our model stipulates a PKI with a single CA. This could be a limitation for the analysis of scenarios with multiple CAs. However, as we argue next, implicitly we do capture at least those multi-CA settings where CAs issue certificates fully independently of each other (as do the CAs of the PKI currently in place to secure web traffic in the Internet) and we believe that our approach hence reaches the majority of practically relevant scenarios. Concretely, verification procedures in independently working multi-CA settings can be readily emulated in our single-CA framework as follows: if $\mathsf{VP}_1, \ldots, \mathsf{VP}_n$ denote the verification procedures of n independent CAs, then composed verification procedure $\mathsf{VP}_{1\ldots n}$ that outputs 1 if there exists at least one i, $1 \leq i \leq n$, such that VP_i outputs 1, effectively folds the n procedures into a single one, adequately modelling verification in the multi-CA setting. Note that we allow an individual CA to maintain state, such as a list of certificates that it has already issued, but we do not assume any joint state between multiple CAs. We remark that the situation in a multi-CA setting where CAs interact is much more involved. For instance, lifting the CA behaviours suggested in Examples 1(b) and (c) to a setting with multiple CAs requires the latter to consistently operate on specific global data structures; this seems inherently hard to implement (as, amongst others, it would implicitly solve the consensus problem in distributed systems).

Remark 2 (One key for many identifiers) A frequently observed setting in the web PKI is that users hold one public key that they bind to a set of identifiers (with the corresponding number of certificates). This happens, for instance, if globally acting companies register domain names in several different countries, but process HTTPS requests on a single centralized facility. Observe that our model covers such cases of multiple registration of a key: hregister may be queried many times for the same pk. Moreover, through being identified by pairs (C, i) (as opposed to (pk, i)), sessions associated with one pk are aware of the identifier $C.\mathsf{id}$ in use.

The queries in set $\mathcal{Q}_S = \{\mathsf{randomness}, \mathsf{session\text{-}key}, \mathsf{corrupt}\}$ model the corruption of a user's secrets. Similar queries are found in other standard AKE models [5,14].

- corrupt (pk) The query requires $\mathsf{pk} \in \mathcal{HK}$; otherwise, it returns $\perp$. This query returns the secret key sk corresponding to public key pk.
- randomness (s) The query requires that session s exist; otherwise, it returns $\perp$. The query returns the randomness s_{rand}.
- session-key (s) The query requires that session s exist and that $s_{\text{status}} = \texttt{accepted}$; otherwise, it returns $\perp$. The query returns the session key s_{key}.

Remark 3 The randomness query gives the adversary access to the randomness s_{rand} that is used in a particular session. Formally, one may think of the session as an instance of a probabilistic Turing machine: the query gives the adversary the values that are read from the random tape. Practically, this models a randomness generator whose values are revealed after they are produced, for example, by a side-channel attack. This is similar to the ephemeral key reveal query from [32].

The hregister query introduced above only allows registration of keys $\mathsf{pk} \in \mathcal{HK}$, i.e., keys held by honest users. In contrast, the adversary can obtain certificates on *arbitrary* (valid) public keys using the following pkregister query. Going even further, the npkregister query allows registration of objects that are not even public keys (always assuming that VP outputs 1 on the candidate object). These queries will allow modelling Kaliski's attack on MQV [28] and small subgroup attacks [34], amongst others. We emphasize that the queries in set $\mathcal{Q}_R = \{\mathsf{pkregister}, \mathsf{npkregister}\}$ have no counterparts in standard definitions of key exchange security.

- pkregister(pk, $\hat{P}$) The query requires that $\mathsf{pk} \in \mathbf{PK}$ and that VP outputs 1 on input pk and $\hat{P}$; otherwise, it returns $\perp$. The public key pk is registered at the CA for identifier $\hat{P}$. The resulting certificate C is added to the global set of certificates and to the set of certificates generated through pkregister query: $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$ and $\mathcal{C}_{\mathsf{pk}} \leftarrow \mathcal{C}_{\mathsf{pk}} \cup \{C\}$. The query returns C.
- npkregister(pk, $\hat{P}$) The query requires that $\mathsf{pk} \notin \mathbf{PK}$ and that VP outputs 1 on input pk and $\hat{P}$; otherwise, it returns $\perp$. The public key pk is registered at the CA for the identifier $\hat{P}$. The resulting certificate C is added to the global set of certificates and to the set of certificates generated through npkregister query: $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$ and $\mathcal{C}_{\mathsf{npk}} \leftarrow \mathcal{C}_{\mathsf{npk}} \cup \{C\}$. The query returns C.

Remark 4 (Efficiency of pkregister and npkregister processing) Observe that pkregister and npkregister queries require a membership test for set **PK**. In some settings, such tests might correspond to computationally hard problems like quadratic residuosity or DDH.[6] We discuss two problems that might arise when analysing key exchange protocols where **PK**-membership is difficult to assess: (1) If the queries, by whatever means, correctly decide membership in **PK**, then

[6] For instance, consider an RSA modulus N and the (cyclic) group QR_N of quadratic residues modulo N. As the CDH problem in QR_N is provably as hard as factoring N [35,42], it is conceivable to instantiate a DL-based key agreement protocol (like UM, HMQV, etc.) with that group. This would result in $\mathbf{PK} = QR_N$, and pkregister and npkregister queries would have to perform membership tests for this set. However, the latter is assumed to be a hard problem, by the quadratic residuosity assumption [26].

the adversary implicitly gets access to a decision oracle for that set; this has to be taken into account in corresponding security reductions, e.g., when selecting appropriate hardness assumptions. (2) If an analysed ASICS protocol Π is used as a component in a higher-level construction Π', then any security argument that reduces the security of Π' to the security of Π will inevitably have to specify how $\mathsf{pkregister}$ and $\mathsf{npkregister}$ oracles are to be simulated. As mentioned before, in some settings this seems to be infeasible.

Example 1 Here, we formalize some verification procedures that reflect practically relevant checks. Note that the algorithms can access the current state of game state variables, such as $\mathcal{C}, \mathcal{C}_{\mathsf{pk}}, \mathcal{C}_{\mathsf{npk}}$. We use q to denote the query that invoked VP.

(a) *Identity validation* CAs may verify the identity of users, for example by checking their passports or looking up public information of companies. In practice, this means that the adversary cannot register a key for an honest party. This enforces a split between honest users that generate their keys randomly by using $\mathsf{hregister}$, and dishonest users that can generate their keys in various ways, and use $\mathsf{pkregister}$ and $\mathsf{npkregister}$. We model this behaviour by abstracting from the concrete passport checking and instead give an abstract verification procedure whose behaviour mimics the concrete validation behaviour.

$$\mathsf{VP}(\mathsf{pk}, \hat{P}) = \begin{cases} 1 \text{ if } (q = \mathsf{hregister} \wedge \neg\exists C \in \mathcal{C}_{\mathsf{pk}} \cup \mathcal{C}_{\mathsf{npk}} : C.\mathsf{id} = \hat{P}) \\ 1 \text{ if } (q \in \mathcal{Q}_R \ \wedge \ \neg\exists C \in \mathcal{C}_h : C.\mathsf{id} = \hat{P}) \text{ , and} \\ 0 \text{ otherwise.} \end{cases}$$

The above verification procedure prevents the adversary from registering a key for identifier $\hat{P}$ via a query in $\mathcal{Q}_R$ if there already exists a certificate returned as response to a query $\mathsf{hregister}$ with identifier $\hat{P}$. Similarly, VP prevents the adversary from registering a key for identifier $\hat{P}$ via the query $\mathsf{hregister}$ if there already exists a certificate returned as response to a query in $\mathcal{Q}_R$ with identifier $\hat{P}$. This splits the identifiers into two disjoint sets and reflects strong identity validation where impersonation of honest users is impossible. Note that the above abstract VP model is implemented by the actual identity validation.

(b) *Uniqueness of the public key*

$$\mathsf{VP}(\mathsf{pk}, \hat{P}) = \begin{cases} 1 \text{ if } \neg\exists C \in \mathcal{C} : C.\mathsf{pk} = \mathsf{pk} \text{ , and} \\ 0 \text{ otherwise.} \end{cases}$$

The verification procedure takes as input pk, $\hat{P}$ and the global set of certificates $\mathcal{C}$. If there is no certificate in $\mathcal{C}$ which contains public key pk for which a certificate is being requested, then it returns 1. Otherwise it returns 0.

(c) *No two public keys for one identifier*

$$\mathsf{VP}(\mathsf{pk}, \hat{P}) = \begin{cases} 1 \text{ if } \neg\exists C \in \mathcal{C} : C.\mathsf{id} = \hat{P} \text{ , and} \\ 0 \text{ otherwise.} \end{cases}$$

The verification procedure takes as input pk, $\hat{P}$ and the global set of certificates $\mathcal{C}$. If there is no certificate in $\mathcal{C}$ which contains identifier $\hat{P}$, then it returns 1. Otherwise it returns 0.

(d) *No checks at all*

$$\mathsf{VP}(\mathsf{pk}, \hat{P}) = 1 \quad \text{(for all } \mathsf{pk} \text{ and all } \hat{P}).$$

Remark 5 (Certificate signing requests (CSRs)) Some CAs may require users to self-sign the data that is to be certified and to submit the resulting certificate signing request. To model CSRs and related checks, one needs to introduce an additional parameter ϕ in the definitions of the queries $\mathsf{hregister}$, $\mathsf{pkregister}$, and $\mathsf{npkregister}$. Then, the verification procedure of a CA checking CSRs takes as input pk, $\hat{P}$ and a signature ϕ. If ϕ is a valid signature on message $m = \hat{P} \parallel \mathsf{pk}$, then it returns 1. Otherwise, it returns 0.

2.2 Security experiment

Using the above queries, we define a parameterized family of AKE security models. As is common in BR-style AKE models, we must restrict query usage so that the adversary cannot trivially win the security experiment. The conditions under which queries are disallowed are expressed by a *freshness* condition, which typically uses a *matching* condition to formalize intended partner sessions.

Definition 2 (Matching, freshness, ASICS model) Let Π be an ASICS protocol. A *matching condition* M for Π is a binary relation on the set of sessions of Π. Let Q be a set of queries such that $\mathcal{Q}_N \subseteq Q \subseteq \mathcal{Q}_N \cup \mathcal{Q}_S \cup \mathcal{Q}_R$. A *freshness condition* F for (Π, Q) is a predicate (usually depending on a matching condition M) that takes a session of Π and a sequence of queries (including arguments and results) of a security experiment over queries in Q. We call a triplet $X = (M, Q, F)$ an *ASICS model* for Π.

Definition 3 gives two possible matching conditions. Examples of freshness conditions will be given in Example 2 on the following page, and in Sect. 4.1.

The intricacies of matching definitions in AKE protocols are explored in detail by Cremers [18]. Two issues are important here. First, there is a strong connection between the information used in a matching definition and the information used to compute the session key. Second, some protocols like the two-message versions of MQV and HMQV allow sessions to compute the same key even if they perform

the same role, whereas other protocols such as NAXOS require the sessions that compute the same key to perform different roles. In the remainder of the paper, we will use one of the definitions below, depending on the type of protocol.

Definition 3 (M1-matching, M2-matching) Let s and s' denote two sessions of an ASICS protocol. We say that session s' M1-*matches* (or is M1-*matching*) session s if $s_{\text{status}} = s'_{\text{status}} = \texttt{accepted}$ and

$$(s_{\text{acert}}.\mathsf{pk}, s_{\text{acert}}.\mathsf{id}, s_{\text{pcert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id}, s_{\text{sent}}, s_{\text{rcvd}}) = (s'_{\text{pcert}}.\mathsf{pk}, s'_{\text{pcert}}.\mathsf{id}, s'_{\text{acert}}.\mathsf{pk}, s'_{\text{acert}}.\mathsf{id}, s'_{\text{rcvd}}, s'_{\text{sent}}).$$

Similarly, we say that session s' M2-*matches* (or is M2-*matching*) session s if s' M1-matches session s and $s_{\text{role}} \neq s'_{\text{role}}$.

Thus, we will use M1 for protocols such as the two-message versions of MQV and HMQV, and we will use M2 for protocols such as NAXOS.

Remark 6 (Comparison to matching definition if only one key per identifier is allowed) Note that when users are allowed to have multiple public keys, matching sessions (that is, intended communication partners) of some AKE protocols will not always compute the same session key if the definition of matching does not require agreement on the public keys used in the respective sessions.

The goal of the adversary is to distinguish the session key of a fresh session from a completely random string. This is modelled through an additional query:

– test-session (s) This query requires that session s exists and that $s_{\text{status}} = \texttt{accepted}$; otherwise, it returns $\perp$. A bit b is chosen at random. If $b = 1$, then s_{key} is returned. If $b = 0$, a random element of $\{0,1\}^k$ is returned.

Definition 4 (ASICS_X experiment) Let Π be an ASICS protocol and $X = (M, Q, F)$ be an ASICS model for Π. We define experiment ASICS_X between an adversary E and a challenger who implements all users and the CA as follows:

1. The experiment is initialized with domain parameters for security parameter k.
2. The adversary E can perform any sequence of queries from Q.
3. At some point in the experiment, E issues a test-session query for a session s that has accepted and satisfies F at the time the query is issued.
4. The adversary may continue with queries from Q, under the condition that the test session must continue to satisfy F.
5. Finally, E outputs a bit b' as E's guess for test-session's bit b.

Definition 5 (ASICS_X advantage) The adversary E wins the ASICS_X security experiment if its output matches the bit b chosen in the test-session query. The ASICS_X-*advantage* of E is defined as $\mathrm{Adv}^{\mathsf{ASICS}_X}_{\Pi,E}(k) = |2\Pr(b = b') - 1|$.

Definition 6 (ASICS security) Let Π be an ASICS protocol and $X = (M, Q, F)$ be an ASICS model. Π is said to be *secure in ASICS model* X if, for all PPT adversaries E, it holds that

1. if two users successfully accept in M-matching sessions, then they both compute the same session key, and
2. E has no more than a negligible advantage in winning the ASICS_X experiment; that is, there exists a negligible function negl in the security parameter k such that $\mathrm{Adv}^{\mathsf{ASICS}_X}_{\Pi,E}(k) \leq \mathrm{negl}(k)$.

Remark 7 (Implicit authentication) Note that the ASICS security definition, like eCK-style security definitions, only provides implicit peer authentication, meaning that the key could only be known by the peer, not explicit authentication that the peer actually was active in the session.

Remark 8 (Comparison to other models, incl. BR, CK and eCK) By choosing the appropriate subset of queries and definitions of freshness and matching, our parameterized model can be instantiated to produce many existing AKE models. The vast majority of existing models assume honestly generated key pairs that are securely distributed to users. They therefore lack pkregister and npkregister queries. Our model family includes the eCK model [32] and the eCK^w model [19] as special cases. These models capture the leakage of session-specific ephemeral data. It also includes the eCK-PFS model [19] incorporating perfect forward secrecy. The Bellare–Rogaway model [5] can be captured in our approach by including timestamps as part of the messages, to accurately model the matching definition from [5]. Unlike the CK model [14], we do not assume unique session identifiers to be available to the protocol up-front. We do not consider this a drawback as real-world protocols usually do not have such identifiers either. Instead, our family includes the basic CK_{HMQV} model [30] in which the HMQV protocol was analysed. For the CK and CK_{HMQV} models, we have chosen to interpret the session state to be the session randomness, in the same fashion as the models have been used by their original authors [14,30].

Remark 9 (Freshness depends on the set of allowed queries and on the matching definition) Note that the exact definition of freshness F used in a security model X will likely depend essentially on the set of queries Q available to the adversary

in that model, as well as the specific matching definition. For example, when Q includes pkregister and/or npkregister queries, then we will require an additional restriction on the registration status of the specific public key of the peer used in the test session.

Example 2 Let us consider the following ASICS model as a concrete example. Let $X = (\mathsf{M1}, Q, F)$ be the ASICS model given by $Q = \mathcal{Q}_N \cup \{\mathsf{session\text{-}key}\} \cup \mathcal{Q}_R$ and F defined as follows. Given a session s and a sequence of queries, F holds if:

- no $\mathsf{session\text{-}key}(s)$ query has been issued, and
- for all sessions s' such that s' M1-matches s, no query $\mathsf{session\text{-}key}(s')$ has been issued, and
- no query $\mathsf{pkregister}(s_{\text{pcert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id})$ has been issued.

The model X is an extension of a BR-like model with a CA that allows registration of arbitrary keys. If a protocol is secure in X, then it is secure even if the adversary can register arbitrary bitstrings as public keys, as long as the specific peer key used in the test session is not generated by the adversary.

2.3 Capturing attacks

We illustrate how several attacks exploiting the adversary's ability to register valid or invalid public keys can be captured in ASICS models.

Kaliski's online UKS attack against MQV [28] Basing on the observations from Example 2, Kaliski's attack against MQV is captured by an ASICS model where the adversary can register a specific (valid) public key with his own identifier via a pkregister query. As the adversary knows the secret key corresponding to the registered public key, the attack cannot be prevented by VP requiring a proof-of-possession of the secret key.

UKS attack against KEA based on public key re-registration [33, p. 380] Suppose that public key pk has been honestly registered at the CA for some user with identifier $\hat{P}$ via an $\mathsf{hregister}(\mathsf{pk}, \hat{P})$ query. In the UKS attack on the KEA protocol, the adversary re-registers the public key pk under his own identifier $\hat{L} \neq \hat{P}$ by issuing the query $\mathsf{pkregister}(\mathsf{pk}, \hat{L})$. The attack is prevented if VP checks for uniqueness of the public key and outputs 0 when the public key was certified before (as observed in [33, p. 381]). Note that the UKS attack can also be prevented by making the session key derivation depend on users' identifiers.

UKS attack against KEA+ based on impersonation attack during key registration Lauter and Mityagin [33] produced the KEA+ protocol from the KEA protocol and Protocol 4 in [7] by incorporating the identifiers of the user and its peer in the session key computation to prevent UKS attacks; however, a similar UKS attack first identified in the current paper still works on the KEA+ protocol. This attack involves (a) a type of impersonation during key registration [45, p. 3] as it requires the adversary to successfully impersonate a user to the CA who then issues a certificate containing the user's identifier, but the adversary's valid public key, and (b) impersonation of that user during the key exchange phase. We stress that the attack does not arise when only one public key per identifier can be registered.

Our UKS attack against KEA+ is captured in the ASICS model $Z = (\mathsf{M2}, Q, F)$ given by $Q = \mathcal{Q}_N \cup \{\mathsf{session\text{-}key}\} \cup \{\mathsf{pkregister}\}$ and F defined as follows. We say that a session s satisfies F if it holds that

- no $\mathsf{session\text{-}key}(s)$ query has been issued, and
- for all sessions s' such that s' M2-matches s, no query $\mathsf{session\text{-}key}(s')$ has been issued, and
- no query $\mathsf{pkregister}(s_{\text{pcert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id})$ has been issued.

We next show that KEA+ is insecure in ASICS model Z. The adversary creates an initiator session via the query $\mathsf{create}(s = (C, i), \mathcal{I}, C')$, where the public keys $C.\mathsf{pk} = A = g^a$ and $C'.\mathsf{pk} = B = g^b$ were honestly registered via the query hregister. Let $C.\mathsf{id} = \hat{A}$. Now the adversary registers a second public key $A' = A^r$, for some r in $\mathbb{Z}_p$, with identifier $\hat{A}$, by issuing the query $\mathsf{pkregister}(A', \hat{A})$ returning certificate C''. He then creates a responder session via the query $\mathsf{create}(s' = (C', j), \mathcal{R}, C'')$ and activates session s' by sending the message $X = g^x$ sent by session s to session s'. The adversary intercepts the response $Y = g^y$ sent by session s', modifies it by setting $Y' = Y^r$, and then sends message Y' to session s. Session s accepts the key $s_{\text{key}} = H(Y'^a, B^x, \hat{A}, \hat{B})$ as the session key, while session s' accepts as its key $s'_{\text{key}} = H(A'^y, X^b, \hat{A}, \hat{B})$. The completed session s is chosen as the test session. Now a session-key query to session s' reveals the session key of the test session. The sessions s and s' are not M2-matching (since $s_{\text{rcvd}} \neq s'_{\text{sent}}$), but compute the same session key. This leads to our UKS attack. Adding either the long-term public keys of both actor and peer or the exchanged messages (in addition to the identifiers) in the session key computation would prevent the attack, since then the session keys computed in both sessions would be different with overwhelming probability (assuming that H is modelled as a random oracle). Alternatively, the attack can be prevented by VP requiring identity validation as described in Example 1; this verification procedure prevents the adversary from registering a key for identifier $\hat{A}$ via the query pkregister as there already exists a certificate returned as response to the query hregister with identifier $\hat{A}$. Note that if KCI attacks are permitted in the

model, the UKS attack cannot be prevented by VP requiring a proof-of-possession of the secret key corresponding to the public key A'.

Online UKS attack on STS-MAC based on duplicate-signature key selection (DSKS) [9] Suppose that the signature scheme employed in the STS-MAC protocol is vulnerable to DSKS attacks. Note that resistance to DSKS attacks is not covered by standard security notions for signatures such as EUF-CMA or SEUF-CMA security, which concern only single keys. The UKS attack on STS-MAC [9, p. 160] exploits the ability of the adversary to register a valid public key pk with known secret key under his own identifier during the run of the protocol. More precisely, the adversary first intercepts a user's message containing a signature σ on message m. He then issues a query pkregister(pk, $\hat{L}$) such that σ is also a valid signature on m under pk. The query associates pk with the adversary's identifier $\hat{L}$. Since the adversary knows the secret key corresponding to pk, he obtains a certificate from the CA even if VP requires a proof-of-possession. Countermeasures to such UKS attacks via modification of the protocol are available [9].

Lim–Lee-style attack against HMQV with special domain parameters, without validation of ephemeral public keys [37] Suppose that the underlying group G is a subgroup of the group of units $U(R)$ of a ring R containing an element $T \neq 0$ such that $T^2 = 0$. Then, the HMQV protocol is vulnerable to the attack described in [37, p. 137] with respect to an ASICS model in which the adversary is only given access to standard protocol execution queries in the set $\mathcal{Q}_N$. However, as mentioned in [37, p. 134], this attack on HMQV can only be launched in certain exotic groups that have never been considered for practical use. The attack can be eliminated by restricting the group used for analysing HMQV in some way or by adding group membership tests on ephemeral public keys to the protocol.

Lim–Lee-style attack against HMQV with DSA domain parameters, without validation of ephemeral public keys [36] Let $G = \langle g \rangle$ denote a q-order subgroup of $\mathbb{Z}_p^*$, where q and p are prime and $(p-1)/q$ is smooth (i.e., it contains no large prime factor). The attack on two-pass HMQV [36, p. 5] can be captured in an ASICS model where the adversary is given access to at least the queries in the set $Q = \mathcal{Q}_N \cup (\mathcal{Q}_S \backslash \{\mathsf{corrupt}\}) \cup (\mathcal{Q}_R \backslash \{\mathsf{pkregister}\})$. In particular, the attack requires that the adversary can register invalid public keys via the npkregister query and reveal the randomness and the session keys via the randomness and session-key queries. This attack can be prevented by countermeasures such as requiring VP to include a group membership test on the public key submitted for certification, or by including group membership tests on both ephemeral and long-term public keys during protocol execution. Small subgroup attacks may also exist in other settings, for instance in groups induced by elliptic curves.

We revisit some of the previously mentioned attacks in Sect. 4.2 to illustrate that the modular approach that we develop in Sect. 3 cannot be applied to the corresponding protocols.

3 Achieving ASICS security

We provide a modular approach to obtain provable ASICS security for certain types of protocols. We first show in Theorem 1 how a result from Kudla and Paterson [31, Theorem 2] can be adapted to incorporate adversarial registration of valid public keys. Then, in Theorem 2, we indicate how to transform protocols to achieve security in the presence of adversaries that can register arbitrary invalid public keys. We start by defining an adapted version of *strong partnering* [31].

Definition 7 (Strong partnering) Let Π be an ASICS protocol, and let $X = (M, Q, F)$ be an ASICS model. We say that Π has *strong partnering in the* ASICS_X *experiment* if no PPT adversary, when attacking Π in the ASICS_X experiment, can establish two sessions s and s' of protocol Π holding the same session key without being M-matching, with more than negligible probability in the security parameter k.

We next define an adapted version of the *session string decisional problem* [31] for an ASICS protocol.

Definition 8 (Session string decisional problem) Let Π be an ASICS protocol and let $X = (M, Q, F)$ be an ASICS model. The session string decisional problem for Π is the algorithmic problem of deciding, for an arbitrary session s with $s_{\text{status}} = \texttt{accepted}$ in an ASICS_X experiment and an arbitrary string m, whether or not $m = ss$, where ss is the session string of session s.

Given an ASICS model $X = (M, Q, F)$, we denote by cNR-X (using terminology from [31], where *cNR* stands for "computational No-Reveals", referring to the absence of session key reveals) the reduced *computational* ASICS_X experiment which is similar to the ASICS_X experiment except that the adversary (a) is not allowed to issue session-key and test-session queries, (b) must pick a session that has accepted and satisfies F at the end of its execution, and (c) output the session key for this session. See Kudla and Paterson [31] for a more detailed description of reduced games.

Definition 9 (cNR-X security) Let Π be an ASICS protocol and $X = (M, Q, F)$ be an ASICS model. Π is said to be cNR-X-*secure* if, for all PPT adversaries E, it holds that

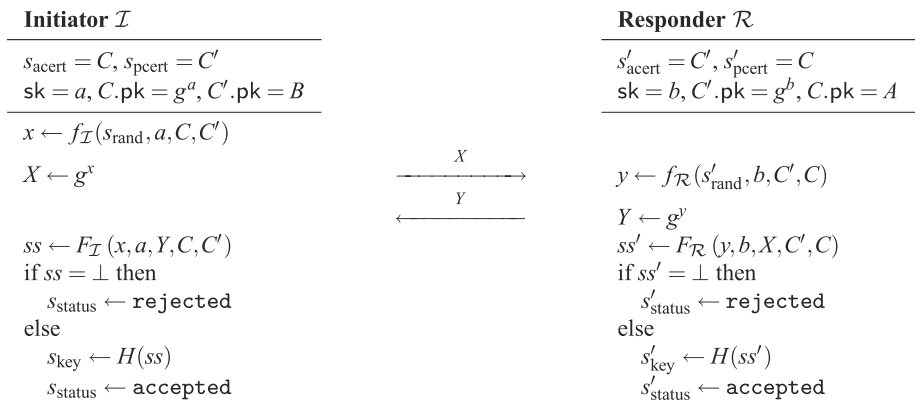

Fig. 1 Messages for generic DH-type ASICS protocol

1. if two users successfully accept in M-matching sessions, then they both compute the same session key, and
2. E has no more than a negligible advantage in winning the cNR-X experiment; that is, there exists a negligible function negl in the security parameter k such that $\text{Adv}^{\text{cNR-}X}_{\Pi,E}(k) \leq \text{negl}(k)$, where $\text{Adv}^{\text{cNR-}X}_{\Pi,E}(k)$ is defined as the probability that E outputs (s, s_{key}) for a session s that has accepted and satisfies F.

Our first theorem deals with the security of DH-type ASICS protocols, which are a generalization of DH-type AKE protocols of Cremers and Feltz [19] to include certificates and to explicitly identify session strings. This class of protocols includes the most prominent modern two-message AKE protocols.

Definition 10 (DH-type ASICS protocol) A *DH-type ASICS protocol* is an ASICS protocol of the following form, specified by functions $f_{\mathcal{I}}$, $f_{\mathcal{R}}$, $F_{\mathcal{I}}$, $F_{\mathcal{R}}$, H:

- Domain parameters (G, g, q), where $G = \langle g \rangle$ is a group of prime order q generated by g.
- KeyGen(): Choose $a \in_R [0, q-1]$. Set $A \leftarrow g^a$. Return secret key $\mathsf{sk} = a$ and public key $\mathsf{pk} = A$.
- $\mathsf{VP}(x, \hat{P}) = 1$ for all x and all $\hat{P}$ (i.e., the CAs do not perform any checks).
- When the initiator is activated to create a new session s with a $\mathsf{create}(s = (C, i), r = \mathcal{I}, C')$ query, where the secret key corresponding to $C.\mathsf{pk}$ is a, the initiator computes an outgoing ephemeral public key $X \leftarrow g^{f_{\mathcal{I}}(s_{\text{rand}}, a, C, C')}$ and returns X as an outgoing message.
- The responder is activated to create a new session s' with a $\mathsf{create}(s = (C', j), r = \mathcal{R}, C)$ query, where the secret key corresponding to $C'.\mathsf{pk}$ is b.
- When the responder is activated in a session s' with a $\mathsf{send}(s', M = X)$, the responder computes an outgoing ephemeral public key $Y \leftarrow g^{f_{\mathcal{R}}(s'_{\text{rand}}, b, C', C)}$ and returns Y as an outgoing message. The responder computes a *session string* (an intermediate value to which we give a special name)

$$ss' \leftarrow F_{\mathcal{R}}(f_{\mathcal{R}}(s'_{\text{rand}}, b, C', C), b, X, C', C),$$

  a session key $s'_{\text{key}} \leftarrow H(ss')$, and accepts by setting $s'_{\text{status}} \leftarrow$ accepted.
- When the initiator is activated in a session s with a $\mathsf{send}(s, M = Y)$, the initiator computes a session string $ss \leftarrow F_{\mathcal{I}}\left(f_{\mathcal{I}}(s_{\text{rand}}, a, C, C'), a, Y, C, C'\right)$, a session key $s_{\text{key}} \leftarrow H(ss)$, and accepts by setting $s_{\text{status}} \leftarrow$ accepted. The structure of protocols of this type is also clarified in Fig. 1.

Theorem 1 *Let $X = (M, Q, F)$ be an ASICS model with $\mathcal{Q}_N \subseteq Q \subseteq \mathcal{Q}_N \cup \mathcal{Q}_S$. Let $Y = (M, Q', F')$ be the ASICS model where $Q' = Q \cup \{\mathsf{pkregister}\}$ and F' is defined as follows. A session s is said to satisfy F' if it satisfies F and no $\mathsf{pkregister}(s_{\text{pcert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id})$ query has been issued. Let Π be a DH-type ASICS protocol. Suppose that*

- *Π has strong partnering in the ASICS_Y experiment,*
- *cNR-X security of the related protocol π (defined in the same way as Π except that the session key generated in π is the session string of Π (i.e., $s^{\pi}_{\text{key}} = ss^{\Pi}$)) is probabilistic polynomial-time reducible to the hardness of the computational problem of some relation ϕ,*
- *the session string decisional problem in the ASICS_Y experiment for Π is polynomial-time reducible to the decisional problem of ϕ, and*
- *there is a polynomial-time algorithm that decides whether an arbitrary bitstring is an element of G,*

then the security of Π in ASICS model Y is probabilistic polynomial-time reducible to the hardness of the gap problem of ϕ, if H is modelled as a random oracle.

In the cNR-X experiment of Theorem 1 the queries session-key and pkregister are not allowed, whereas in ASICS_Y both queries are allowed. Theorem 1 states that for any DH-type protocol Π, under certain conditions, it holds that security of the related protocol π in a reduced model (in which public keys can only be honestly registered) implies

security of Π in the stronger non-reduced model that additionally captures adversarial registration of valid public keys.

The following theorem, which is applicable to a wider range of protocols than Theorem 1 (e.g., to three-message protocols such as UM [38] or HMQV-C [29]), allows us to achieve security against adversaries that can obtain certificates from the CA for invalid public keys by transforming the protocol to include a group membership test on the peer's public key. In contrast to Theorem 1, no additional requirement is imposed on the freshness condition of model Y.

Theorem 2 *Let $X = (M, Q, F)$ be an ASICS model with $\mathcal{Q}_N \subseteq Q \subseteq \mathcal{Q}_N \cup \mathcal{Q}_S \cup (\mathcal{Q}_R \backslash \{\mathsf{npkregister}\})$.*

Let Π be an ASICS protocol where the domain parameters (G, g, q), the key generation algorithm KeyGen *and the verification procedure* VP *are as in Definition 10.*

Let $f(\Pi)$ denote the ASICS protocol derived from Π by adding the following protocol step for each role of the protocol. Upon creation with (or, via send*, receipt of) the certificate C' to be used for the peer of session s, the user running session s checks whether the public key $C'.\mathsf{pk}$ belongs to the group G before continuing the execution of the protocol. In case the check fails, the protocol execution is aborted and s_{status} is set to* `rejected`.

Suppose that protocol Π is secure in ASICS model X and that there is a polynomial-time algorithm that decides whether an arbitrary bitstring is an element of G. Then, the transformed protocol $f(\Pi)$ is secure in ASICS model $Y = (M, Q \cup \{\mathsf{npkregister}\}, F)$.

The relative strengths of security between game-based security models were investigated by Choo et al. [17], and formally defined by Cremers and Feltz [20] as follows. Let $\text{secure}(X, \Pi)$ be a predicate that is true if and only if the protocol Π is secure in security model X.

Definition 11 [20] Let π be a class of AKE protocols. Let X and Y be two security models. We say that model Y is *at least as strong as* model X with respect to π, denoted by $X \preceq^{\pi}_{\text{Sec}} Y$, if

$$\forall\, \Pi \in \pi.\ \text{secure}(Y, \Pi) \Rightarrow \text{secure}(X, \Pi). \tag{1}$$

We say that model Y is *stronger* than model X with respect to protocol class π, if $X \preceq^{\pi}_{\text{Sec}} Y$ and not $Y \preceq^{\pi}_{\text{Sec}} X$.

Definition 12 Let π be a class of AKE protocols. We say that two security models X and Y are *equally strong* with respect to π, denoted by $X =^{\pi}_{\text{Sec}} Y$, if $X \preceq^{\pi}_{\text{Sec}} Y$ and $Y \preceq^{\pi}_{\text{Sec}} X$, where the relation $\preceq^{\pi}_{\text{Sec}}$ is as in Definition 11.

Remark 10 Let π be the class of ASICS protocols where the domain parameters (G, g, q), the key generation algorithm KeyGen are as in Definition 10 and the verification procedure $\mathsf{VP}(x, \hat{P})$ outputs 1 if $x \in G$ and 0 otherwise. Let $X = (M, Q, F)$ with $\mathcal{Q}_N \subseteq Q \subseteq \mathcal{Q}_N \cup \mathcal{Q}_S \cup (\mathcal{Q}_R \backslash \{\mathsf{npkregister}\})$ and $Y = (M, Q \cup \{\mathsf{npkregister}\}, F)$. It is easy to verify that the ASICS models X and Y are equally strong with respect to π according to Definition 12. Whenever the adversary issues a query npkregister in an ASICS_Y experiment, the symbol $\perp$ is returned. Thus, transforming an ASICS protocol from protocol class π as described in Theorem 2 does not yield a protocol that is secure in a stronger ASICS model. In contrast, the model Y is stronger than model X with respect to the protocol class considered in Theorem 2.

Combining both theorems, we obtain the following result.

Corollary 1 *Let Π be a DH-type ASICS protocol. Let $X = (M, Q, F)$ and $Y = (M, Q', F')$ be defined as in Theorem 1, and let the conditions of Theorem 1 hold with respect to protocol Π. Let $f(\Pi)$ denote the protocol derived from Π as specified in Theorem 2. Then, the transformed protocol $f(\Pi)$ is secure in ASICS model $Z = (M, Q'', F')$, where $Q'' = Q' \cup \{\mathsf{npkregister}\}$, if H is modelled as a random oracle.*

Applying Corollary 1 to a concrete DH-type ASICS protocol that satisfies all the preconditions, we obtain a protocol that is secure in an ASICS model in which (a) sessions (including the test session) may use a certificate for the peer that resulted from an npkregister query, and (b) the certificate of the test session's peer was not the result of a pkregister query.

4 Applications

To illustrate the power of our generic approach, we examine in Sect. 4.1 how to apply our technique to Ustaoglu's CMQV protocol [46]. CMQV is a modern DH-type protocol that is comparable in efficiency to HMQV, but enjoys a simpler security proof in the eCK model.

Our results allow us to analyse CMQV in a model that does not include queries session-key, pkregister, and npkregister, which simplifies the overall proof. We verify that CMQV meets the preconditions of Corollary 1 and conclude that a variant of CMQV with group membership test on the peer's public key is ASICS-secure in an eCK-like model. Similarly, our generic approach can be applied to other DH-type candidates such as NAXOS [32] and `UP` [47]. We discuss in Sect. 4.2 where the preconditions of Corollary 1 are violated for protocols from Sect. 2.3 such as MQV or HMQV.

4.1 CMQV

CMQV [46] was originally proven secure in the eCK model, where there is only one public key per identifier. In the ASICS setting, there is no such unique mapping between user identifiers and public keys. Hence, to be able to prove CMQV

Fig. 2 Specification of $f_{\mathcal{I}}$, $f_{\mathcal{R}}$, $F_{\mathcal{I}}$, $F_{\mathcal{R}}$ for CMQV

$$f_{\mathcal{I}}(r,a,C,C') = \mathcal{H}_1(r,a)$$
$$F_{\mathcal{I}}(x,a,Y,C,C') = \begin{cases} \perp & \text{, if } Y \notin G\setminus\{1\} \\ ((YB^e)^{x+da} \parallel g^x \parallel Y \parallel C.\mathsf{id} \parallel A \parallel C'.\mathsf{id} \parallel B) & \text{, if } Y \in G\setminus\{1\} \end{cases}$$

$$f_{\mathcal{R}}(r',b,C',C) = \mathcal{H}_1(r',b)$$
$$F_{\mathcal{R}}(y,b,X,C',C) = \begin{cases} \perp & \text{, if } X \notin G\setminus\{1\} \\ ((XA^d)^{y+eb} \parallel X \parallel g^y \parallel C.\mathsf{id} \parallel A \parallel C'.\mathsf{id} \parallel B) & \text{, if } X \in G\setminus\{1\} \end{cases}$$

secure in the ASICS model, we need to include the public keys of the users in the session string to ensure that they have the same view of these keys when deriving the session key.

CMQV as a DH-type ASICS protocol Two-pass CMQV can be stated as a DH-type ASICS protocol, by instantiating Definition 10 with the following functions. Let $\mathcal{H}_1 : \{0,1\}^k \times \mathbb{Z}_q^* \to \mathbb{Z}_q^*$, $\mathcal{H}_2 : \{0,1\}^* \to \mathbb{Z}_q$, and $H : \{0,1\}^* \to \{0,1\}^k$ be hash functions. We define $f_{\mathcal{I}}$, $f_{\mathcal{R}}$, $F_{\mathcal{I}}$, $F_{\mathcal{R}}$ as in Fig. 2, where $d = \mathcal{H}_2(X \parallel C.\mathsf{id} \parallel C'.\mathsf{id})$, $e = \mathcal{H}_2(Y \parallel C.\mathsf{id} \parallel C'.\mathsf{id})$, $A = C.\mathsf{pk}$, $B = C'.\mathsf{pk}$; $\parallel$ denotes tagged concatenation to avoid ambiguity with variable-length strings.

We now show, using Corollary 1, that the resulting DH-type CMQV protocol is a secure ASICS protocol in an ASICS model with leakage queries corresponding to the eCK model.

ASICS model for eCK-like leakage Define the ASICS model $\mathsf{eCK} = (\mathsf{M2}, Q, F)$ for eCK-like leakage [32] as follows. Let $Q = \mathcal{Q}_N \cup \mathcal{Q}_S$. Let F be the condition that a session s satisfies F if, for all sessions s' such that s' M2-matches s, none of the following conditions hold:

- a $\mathsf{session\text{-}key}(s)$ query has been issued;
- if s' exists:
  - a $\mathsf{session\text{-}key}(s')$ query has been issued;
  - both $\mathsf{corrupt}(s_{\text{acert}}.\mathsf{pk})$ and $\mathsf{randomness}(s)$ have been issued;
  - both $\mathsf{corrupt}(s'_{\text{acert}}.\mathsf{pk})$ and $\mathsf{randomness}(s')$ queries have been issued;
- if s' does not exist:
  - both $\mathsf{corrupt}(s_{\text{acert}}.\mathsf{pk})$ and $\mathsf{randomness}(s)$ have been issued;
  - a $\mathsf{corrupt}(s_{\text{pcert}}.\mathsf{pk})$ query has been issued.

Theorem 3 *Let f(CMQV) be the DH-type ASICS protocol derived from the* CMQV *protocol defined above, as specified in Theorem 2. If $\mathcal{H}_1, \mathcal{H}_2$ and H are modelled as random oracles, G is a group where the gap Diffie–Hellman assumption holds and membership in G is decidable in polynomial time, then f(CMQV) is secure in ASICS model $Z = (\mathsf{M2}, \mathcal{Q}_N \cup \mathcal{Q}_S \cup \mathcal{Q}_R, F')$, where a session s is said to satisfy F' if it satisfies the freshness condition F from the* eCK *model and no* $\mathsf{pkregister}(s_{\text{pcert}}.\mathsf{pk}, s_{\text{pcert}}.\mathsf{id})$ *query has been issued.*

Proof (Sketch) We can readily show that CMQV satisfies the preconditions of Corollary 1 under the above formulation of the eCK model as an ASICS model:

1. *Strong partnering* It is straightforward to see that CMQV has strong partnering in the $\mathsf{ASICS}_{\mathsf{eCK}'}$ game (where eCK' is derived from eCK as described in Theorem 1): since the session key in CMQV is computed via a random oracle, the probability that two sessions derive the same session key without using the same session string input to the random oracle is negligible.
2. cNR-eCK-*security of session string variant of CMQV* This can be shown by an adaptation of Ustaoglu's original proof of CMQV. In large part, the main proof can be followed. However, a few simplifications can be made because the simulation need not answer session-key queries (so preventing key replication attacks and simulating sessions where the public key is a challenge value are easier).
3. *Hardness of the session string decision problem* It can be easily seen that this is polynomial-time reducible to the decisional problem for Diffie–Hellman triples (U, V, W) by noting that the first component of the CMQV session string σ is equal to

$$g^{(y+eb)(x+da)} = g^{xy} g^{ady} g^{bex} g^{abde}.$$

The DDH values (U, V) can be injected into either (X, Y), (A, Y), (B, X), or (A, B), with W inserted into the corresponding part of σ, yielding a polynomial-time reduction.

Detailed proofs of each of the above claims can be found in Appendix 2. □

4.2 Discussion

In Sect. 4.1, we provided evidence that our modular approach can indeed be successfully applied to reason about the security of prominent key exchange protocols. Here, we inspect

which preconditions of Corollary 1 are violated for some commonly analysed protocols.

Let $X_1 = (\mathsf{M1}, Q, F)$ be the ASICS model given by $Q = \mathcal{Q}_N \cup \{\mathsf{session\text{-}key}\}$ and F defined as follows. We say that a session s satisfies F if it holds that neither a $\mathsf{session\text{-}key}(s)$ query nor a $\mathsf{session\text{-}key}(s')$ query for any session s' that is M1-matching s have been issued. Let $Y_1 = (\mathsf{M1}, Q', F')$ be derived from X_1 as specified in Theorem 1. The UKS attacks described in Sect. 2.3 against the DH-type protocols MQV and KEA caused two sessions that were not M1-matching to compute the same session key. Thus, the first precondition of Corollary 1 is violated for these protocols as they do not have strong partnering in the ASICS_{Y_1} experiment. The two-pass HMQV protocol without ephemeral public key validation is insecure in ASICS model X_1 when special domain parameters are used (see Sect. 2.3) due to the lack of ephemeral public key validation. Hence, it is easy to see that the related protocol, that we call HMQV', is not cNR-X_1-secure, which violates the second precondition of Corollary 1. Even though it is unclear whether there is an attack against two-pass HMQV with DSA domain parameters without ephemeral public key validation, the third precondition of Theorem 1 is violated for this version of the HMQV protocol. The lack of ephemeral public key validation hinders us from using a DDH oracle to solve the session string decisional problem for the protocol since there is no guarantee that all inputs to the DDH oracle belong to the group G. The latter observations on HMQV lead us to the conclusion that validation of ephemeral public keys is necessary to be able to apply Corollary 1 to the HMQV protocol, even when the adversary is not given access to the $\mathsf{randomness}$ query.

Let $X_2 = (\mathsf{M2}, Q, F)$ be the ASICS model given by $Q = \mathcal{Q}_N \cup \{\mathsf{session\text{-}key}\}$ and F defined as follows. We say that a session s satisfies F if it holds that neither a $\mathsf{session\text{-}key}(s)$ query nor a $\mathsf{session\text{-}key}(s')$ query for any session s' that is M2-matching s have been issued. Let $Y_2 = (\mathsf{M2}, Q', F')$ be derived from X_2 as specified in Theorem 1. Note that model Y_2 corresponds to the model X defined in Sect. 2.3 capturing the UKS attack against KEA+. The latter UKS attack described against the DH-type protocol KEA+ caused two sessions that were not M2-matching to compute the same session key. Thus, the first precondition of Corollary 1 is violated for KEA+ as this protocol does not have strong partnering in the ASICS_{Y_2} experiment.

5 Lessons learned and recommendations

As we started our systematic investigation, we assumed that certification authorities would need to perform some minimal checks on public keys to obtain secure KE protocols. Perhaps surprisingly, nearly all of the effort can be shifted to the protocols; and modern protocols often perform sufficient checks. In particular, our results provide formal foundations for some of the protocol-specific observations of Menezes and Ustaoglu [37]: checking that short-term as well as long-term public keys are in the key space (i.e., in group G for DH-type protocols) is not superfluous.

Based on these observations, and given M public keys, N users may need to perform on the order of $M \times N$ such checks in total, even when caching the results. Reasoning purely about the overall amount of computation time used, one could consider moving the burden to the CAs. If the CAs only create certificates after a successful check, the CAs would only perform on the order of M checks in total. Depending on the deployment scenario, this might be a preferable alternative.

Similarly, CAs do not necessarily need to check uniqueness of public keys. As long as the key derivation involves the identifiers in an appropriate way, UKS attacks such as the one on KEA can be prevented. Even if the adversary re-registers an honest user's public key as his own public key, ASICS security for the honest user is still guaranteed as long as the adversary does not know the corresponding private key.

In general, our results further justify using as much information as possible in the key derivation function (KDF). This helps with establishing formal security proofs and it is also a prudent engineering principle. In particular, we recommend that in settings where users may have multiple long-term public keys, the input to the KDF should not only include the identifiers and the message transcript, but also the specific public keys used in the session.

We hope our work can serve as a foundation for the development of a range of protocols specifically designed to incorporate certification systems, offering different trade-offs between efficiency and trust assumptions of the involved parties.

Acknowledgments C. B. and D. S. were supported by Australian Research Council (ARC) Discovery Project DP130104304. C. C. was supported by ETH Research Grant ETH-30 09-3. K. G. P. and B. P. were supported by a EPSRC Leadership Fellowship EP/H005455/1. This work was partly done while M. F. was at ETH Zurich, supported by ETH Research Grant ETH-30 09-3.

Appendix 1: Proofs of Theorems 1 and 2

Proof of Theorem 1 The proof structure is similar to the proof of Theorem 2 in [31]. We denote by Λ the session key space associated with protocol Π. Since the cNR-X security of protocol π is probabilistic polynomial-time reducible to the hardness of the computational problem of some relation ϕ, there exists an algorithm A that on input of a problem instance of the computational problem of ϕ and interacting with an adversary E which has non-negligible probability η of winning the cNR-X game for π in time τ is able to solve

the computational problem of ϕ with non-negligible probability $h(\eta)$ and in time $v(\tau)$, for some polynomial functions h and v.

By assumption, the session string decisional problem in the ASICS_Y experiment for Π is polynomial-time reducible to the decisional problem of ϕ. Hence, there is an algorithm W which solves the session string decisional problem for Π in polynomial-time τ'' given access to a decisional oracle for ϕ.

Let D be an adversary winning the ASICS_Y experiment against protocol Π with non-negligible probability η' in time τ'. Let K denote the event that D does not query H with the session string ss^* of the test session s^*. Since Π has strong partnering in the ASICS_Y experiment, it holds that, with overwhelming probability, if two sessions compute the same session key, then they must be M-matching. Thus, if event K occurs, then D can only win the experiment with negligible probability $u(k)+1/|\Lambda|$, where $u(k)$ denotes the probability that D issues a $\mathsf{session\text{-}key}$ query to a session s that is not M-matching s^* and $s_{\mathrm{key}} = s^*_{\mathrm{key}}$.

We next define an algorithm B which solves the gap problem of ϕ with non-negligible probability $h'(\eta')$ and in time $v'(\tau')$, for some polynomial functions h' and v', using adversary D as a subroutine. B will also run algorithm A on the problem instance of the computational problem of ϕ, and an algorithm L that decides, in polynomial-time τ''', whether an arbitrary bitstring pk submitted for certification is an element of G. We now define B's responses to D's queries for the pre-specified peer setting; the post-specified peer case proceeds similarly. Algorithm B maintains sets of certificates $\mathcal{C}_h$ and $\mathcal{C}_{\mathsf{pk}}$ as well as lists H-List and G-List, all of which are initially empty.

1. $q \in Q \cap \{\mathsf{kgen}, \mathsf{randomness}, \mathsf{corrupt}\}$: B forwards the query to A and passes A's response back to D.
2. $\mathsf{hregister}(\mathsf{pk}, \hat{P})$: B forwards the query to A and passes A's response back to D. In case A returns a certificate C, B adds C to the set $\mathcal{C}_h$, i.e., $\mathcal{C}_h \leftarrow \mathcal{C}_h \cup \{C\}$.
3. $\mathsf{pkregister}(\mathsf{pk}, \hat{P})$: B checks whether $\mathsf{pk} \in G$ using algorithm L and whether VP outputs 1 on input pk and $\hat{P}$. If all the checks succeed, then B adds a certificate C to the set $\mathcal{C}_{\mathsf{pk}}$, i.e., $\mathcal{C}_{\mathsf{pk}} \leftarrow \mathcal{C}_{\mathsf{pk}} \cup \{C\}$, and returns C. Else, B returns $\perp$.
4. $\mathsf{create}\left(s = (C, i), r, C'\right)$: B checks whether $C \in \mathcal{C}_h$, a session s with counter i has not yet been created, $r \in \{\mathcal{I}, \mathcal{R}\}$, and $C' \in \mathcal{C}_h \cup \mathcal{C}_{\mathsf{pk}}$. If one of the checks fails, then B returns $\perp$. Else if $C' \in \mathcal{C}_{\mathsf{pk}}$, then B answers D's query by simulating the protocol execution itself. Else, B forwards the query to A and passes A's response (if any) to D.
5. $\mathsf{send}(s, M)$: If session s does not exist or if $s_{\mathrm{status}} \neq \mathtt{active}$, then B returns $\perp$. Else if $s_{\mathrm{pcert}} \in \mathcal{C}_{\mathsf{pk}}$, then B responds to the query by simulating the protocol execution itself. Else B forwards the query to A and passes A's response (if any) to D.
6. H query: To answer D's queries to the random oracle for H, B stores entries of the form (x_i, λ_i) with $\lambda_i \in \Lambda$ in the H-List. When D makes a query x to the random oracle for H, B determines the return value for D as follows:
   - If there exists an entry (x_i, λ_i) in the H-List with $x_i = x$, then return λ_i.
   - Else if there is an entry $(s_{\mathrm{acert}}.\mathsf{id}, s_{\mathrm{acert}}.\mathsf{pk}, s_{\mathrm{pcert}}.\mathsf{id}, s_{\mathrm{pcert}}.\mathsf{pk}, s_{\mathrm{role}}, s_{\mathrm{sent}}, s_{\mathrm{rcvd}}, \lambda_i)$ in the G-List, for some session s that has accepted and $\lambda_i \in \Lambda$, such that x is the session string of session s (i.e., $x = ss$) using algorithm W, then store the entry (x, λ_i) in the H-List and return λ_i.
   - Else, B chooses $\lambda \in_R \Lambda$, stores the entry (x, λ) in the H-List and return λ.
7. $\mathsf{session\text{-}key}(s)$: To answer D's $\mathsf{session\text{-}key}$ queries, B stores entries of the form $(s_{\mathrm{acert}}.\mathsf{id}, s_{\mathrm{acert}}.\mathsf{pk}, s_{\mathrm{pcert}}.\mathsf{id}, s_{\mathrm{pcert}}.\mathsf{pk}, s_{\mathrm{role}}, s_{\mathrm{sent}}, s_{\mathrm{rcvd}}, \lambda_i)$ with $\lambda_i \in \Lambda$ in the G-List. When D makes a $\mathsf{session\text{-}key}(s)$ query to an initiator session s that has accepted, B determines the return value for D as follows:
   - If there exists an entry $(s_{\mathrm{acert}}.\mathsf{id}, s_{\mathrm{acert}}.\mathsf{pk}, s_{\mathrm{pcert}}.\mathsf{id}, s_{\mathrm{pcert}}.\mathsf{pk}, \mathcal{I}, s_{\mathrm{sent}}, , s_{\mathrm{rcvd}}, \lambda_i)$ in the G-List, for some $\lambda_i \in \Lambda$, then return λ_i.
   - Else if there is an entry $(s_{\mathrm{pcert}}.\mathsf{id}, s_{\mathrm{pcert}}.\mathsf{pk}, s_{\mathrm{acert}}.\mathsf{id}, s_{\mathrm{acert}}.\mathsf{pk}, \mathcal{R}, s_{\mathrm{rcvd}}, s_{\mathrm{sent}}, \lambda_i)$ in the G-List, then B stores the entry $(s_{\mathrm{acert}}.\mathsf{id}, s_{\mathrm{acert}}.\mathsf{pk}, s_{\mathrm{pcert}}.\mathsf{id}, s_{\mathrm{pcert}}.\mathsf{pk}, \mathcal{I}, s_{\mathrm{sent}}, s_{\mathrm{rcvd}}, \lambda_i)$ in the G-List and returns λ_i.
   - Else if there exists an entry of the form (x_i, λ_i) in the H-List, where $x_i = ss$ using algorithm W, then B stores the entry $(s_{\mathrm{acert}}.\mathsf{id}, s_{\mathrm{acert}}.\mathsf{pk}, s_{\mathrm{pcert}}.\mathsf{id}, s_{\mathrm{pcert}}.\mathsf{pk}, \mathcal{I}, s_{\mathrm{sent}}, s_{\mathrm{rcvd}}, \lambda_i)$ in the G-List and returns λ_i.
   - Else, B chooses $\lambda \in_R \Lambda$, stores the entry $(s_{\mathrm{acert}}.\mathsf{id}, s_{\mathrm{acert}}.\mathsf{pk}, s_{\mathrm{pcert}}.\mathsf{id}, s_{\mathrm{pcert}}.\mathsf{pk}, \mathcal{I}, s_{\mathrm{sent}}, s_{\mathrm{rcvd}}, \lambda)$ in the G-List, and returns λ.

   A $\mathsf{session\text{-}key}$ query to a responder session that has accepted is answered similarly.
8. $\mathsf{test\text{-}session}(s^*)$: B selects $\mu \in_R \Lambda$ and returns μ to D.
9. D outputs a guess: B aborts with failure.

B can detect the complementary event K^c by checking which of the entries (x_i, λ_i) in the H-List has $x_i = ss^*$ using algorithm W. B then passes x_i to A. Since the test session s^* must be fresh, no $\mathsf{pkregister}(s^*_{\mathrm{pcert}}.\mathsf{pk}, s^*_{\mathrm{pcert}}.\mathsf{id})$ occurred in the ASICS_Y experiment and hence the certificate s^*_{pcert} has been output through an $\mathsf{hregister}(s^*_{\mathrm{pcert}}.\mathsf{pk}, s^*_{\mathrm{pcert}}.\mathsf{id})$ query. A solves the computational problem of ϕ with non-negligible probability $h(\eta)$, where $\eta = \eta'(1 - u(k) - 1/|\Lambda|)$. B is successful by outputting A's solution to the instance of the computational problem of ϕ and solves the gap problem of

ϕ with non-negligible probability $h(\eta)$ and in time $v(\tau)$, where $\tau = \tau' + \tau'' n_H(n_{\mathsf{session\text{-}key}} + 1) + \tau''' n_{\mathsf{pkregister}}$ with n_H, $n_{\mathsf{session\text{-}key}}$ and $n_{\mathsf{pkregister}}$ denoting the number of H, session-key and pkregister queries issued by D, respectively. □

Remark 11 We cannot show that Theorem 1 holds for more complex protocols Π such as UM or HMQV-C in arbitrary ASICS base models as the simulation of non-test sessions s of Π with s_{pcert} being the result of a pkregister query cannot be performed in the appropriate way without the knowledge of long-term secret keys and without violating the freshness condition.

Proof of Theorem 2 Let Π be an ASICS protocol secure in model X. It is straightforward to verify the first condition of Definition 6, that is, that M-matching sessions of protocol $f(\Pi)$ compute the same session key. This follows from the fact that M-matching sessions of protocol Π compute the same key as protocol Π is secure in ASICS model X. We next verify that the second condition of Definition 6 holds.

Claim If there is a PPT adversary E succeeding in the ASICS_Y experiment against protocol $f(\Pi)$ with non-negligible advantage in time τ', then we can construct a PPT adversary E' succeeding in the ASICS_X experiment against protocol Π with non-negligible advantage in time $v(\tau)$ (for some polynomial function v) using adversary E as a subroutine. Let L be an algorithm that decides, in polynomial-time τ'', whether an arbitrary bitstring pk submitted for certification is an element of G.

Proof. Fix a PPT adversary E succeeding in the ASICS_Y experiment against protocol $f(\Pi)$ with non-negligible advantage. We define an algorithm E' which succeeds in the ASICS_X experiment against protocol Π with non-negligible advantage using E as a subroutine. Algorithm E' maintains sets of certificates $\mathcal{C}_h$, $\mathcal{C}_{\mathsf{pk}}$ and $\mathcal{C}_{\mathsf{npk}}$, all of which are initially empty, and answers E's queries in the pre-specified peer setting as follows.

1. $q \in Q \cap \{$kgen, randomness, corrupt, session-key$\}$: E' issues the same query and returns the answer to E.
2. hregister(pk, $\hat{P}$): When E issues an hregister(pk, $\hat{P}$) query, E' issues the same query and returns the answer to E. In case a certificate C is returned, E' adds C to the set $\mathcal{C}_h$, i.e., $\mathcal{C}_h \leftarrow \mathcal{C}_h \cup \{C\}$.
3. pkregister(pk, $\hat{P}$): When E issues a pkregister(pk, $\hat{P}$) query, E' issues the same query and returns the answer to E. In case a certificate C is returned, E' adds C to the set $\mathcal{C}_{\mathsf{pk}}$, i.e., $\mathcal{C}_{\mathsf{pk}} \leftarrow \mathcal{C}_{\mathsf{pk}} \cup \{C\}$.
4. npkregister(pk, $\hat{P}$): E' checks whether pk $\notin G$ (using algorithm L) and VP outputs 1 on input pk and $\hat{P}$. If the checks succeed (i.e., pk $\notin G$ and VP(pk, $\hat{P}$) $= 1$), then E' returns a certificate C to E and adds C to the set $\mathcal{C}_{\mathsf{npk}}$, i.e., $\mathcal{C}_{\mathsf{npk}} \leftarrow \mathcal{C}_{\mathsf{npk}} \cup \{C\}$. Otherwise, E' returns $\perp$.
5. create $\left(s = (C, i), r, C'\right)$: E' checks whether $C \in \mathcal{C}_h$, a session s with counter i has not yet been created, $r \in \{\mathcal{I}, \mathcal{R}\}$, and $C' \in \mathcal{C}_h \cup \mathcal{C}_{\mathsf{pk}} \cup \mathcal{C}_{\mathsf{npk}}$. If one of the checks fails, then E' returns $\perp$. Else if $C' \in \mathcal{C}_h \cup \mathcal{C}_{\mathsf{pk}}$, then E' issues the same query and returns the answer (if any) to E. Else, E' rejects the session creation and sets s_{status} to `rejected`.
6. send (s, M): If session s does not exist or if $s_{\mathrm{status}} \neq$ `active`, then E' returns $\perp$. Else E' issues the same query and returns the response (if any) to E.
7. test-session(s): When E issues a test-session(s) query to a session s that has accepted, E' issues the same test-session query and returns the answer to E.
8. At the end of E's execution, that is, after it has output its guess b', E' outputs b' as well.

It follows that $\mathrm{Adv}^{\mathsf{ASICS}_Y}_{f(\Pi),E}(k) \leq \mathrm{Adv}^{\mathsf{ASICS}_X}_{\Pi,E'}(k)$, and adversary E' runs in time $v(\tau)$ with $\tau = \tau' + \tau'' n_{\mathsf{npkregister}}$, for some polynomial function v, where $n_{\mathsf{npkregister}}$ denotes the number of npkregister queries made by E. Since Π is secure in ASICS model X, $\mathrm{Adv}^{\mathsf{ASICS}_Y}_{f(\Pi),E}(k)$ is bounded above by a negligible function in the security parameter k. □

Appendix 2: Analysis of CMQV

Let $\mathsf{eCK}' = (\mathsf{M2}, Q', F')$ be the ASICS model where $Q' = Q \cup \{$pkregister$\}$ and F' is defined as F with the additional requirement that no pkregister$(s_{\mathrm{pcert}}.\mathsf{pk}, s_{\mathrm{pcert}}.\mathsf{id})$ query has been issued.

Lemma 1 *Let* eCK *and* eCK' *be as above. CMQV has strong partnering in the* $\mathsf{ASICS}_{\mathsf{eCK}'}$ *experiment under the assumption that H is a random oracle.*

Proof Suppose otherwise. Namely, suppose there exists two sessions s and s' of CMQV that hold the same session key but are not M2-matching. Since the session key in CMQV is derived by applying a random oracle, except with negligible probability, the input to the random oracle in both sessions must be the same. Since they are not M2 matching, either $s_{\mathrm{acert}}.\mathsf{id} \neq s'_{\mathrm{pcert}}.\mathsf{id}$, or $s_{\mathrm{acert}}.\mathsf{pk} \neq s'_{\mathrm{pcert}}.\mathsf{pk}$, or $s_{\mathrm{pcert}}.\mathsf{id} \neq s'_{\mathrm{acert}}.\mathsf{id}$, or $s_{\mathrm{pcert}}.\mathsf{pk} \neq s'_{\mathrm{acert}}.pk$, or $s_{\mathrm{sent}} \neq s'_{\mathrm{rcvd}}$, or $s_{\mathrm{rcvd}} \neq s'_{\mathrm{sent}}$, or $s_{\mathrm{role}} = s'_{\mathrm{role}}$.

First suppose $s_{\mathrm{role}} \neq s'_{\mathrm{role}}$. Then, either the public keys, identifiers, or transcripts of the two sessions do not correspond. But these are all inputs to the random oracle, so except with negligible probability the outputs of the random oracle will be different, contradicting that the two sessions hold the same session key.

Now suppose $s_{\mathrm{role}} = s'_{\mathrm{role}}$. Except with negligible probability, two distinct honest sessions will have $s_{\mathrm{rand}} \neq s'_{\mathrm{rand}}$,

and hence $s_{\text{sent}} \neq s'_{\text{sent}}$. But since both s and s' think of themselves as the initiator, they will each put their own sent ephemeral public key in the second component of the call to H, and these values are different, so except with negligible probability the outputs of the random oracle will be different, contradicting that the two sessions hold the same key. □

Let (G, g, q) be as in Definition 10. Let $\phi \subseteq (G \times G) \times G$ be the *Diffie–Hellman relation* on $G = \langle g \rangle$. In particular, (g^a, g^b) is related under ϕ to g^c if and only if $ab \equiv c \mod q$.

Lemma 2 *The* cNR-eCK *security of the variant of CMQV in which the session string is output as the session key is polynomial-time reducible to the computational problem of the Diffie–Hellman relation ϕ, under the assumption that $\mathcal{H}_1$ and $\mathcal{H}_2$ are random oracles.*

The basic idea of the proof is as follows.

- If the adversary happens to figure out a long-term secret key without issuing a corrupt query (event E), it must ask that value to a random oracle $\mathcal{H}_1$, and we can immediately use that value to solve the CDH problem by having embedded one of the CDH challenge values in that public key.
- If the adversary is passive in the test session (event $\overline{E} \wedge M$), we can embed the CDH challenge values U, V as the ephemeral public keys X and Y of the test session. The adversary's view can be simulated perfectly unless the adversary asks either $(\tilde{x}, a)$ or $(\tilde{y}, b)$ as a query for $\mathcal{H}_1$. But the freshness condition prevents the adversary from finding both elements of either pair. Therefore, the adversary cannot do better than guess the session string unless it can compute σ. Here, the CDH of U and V can be extracted from σ.
- If the adversary is active in the test session (event $\overline{E} \wedge \overline{M}$), we can embed the CDH challenge values in the long-term key of the partner of the test session and the ephemeral public key of the session. As before the simulation is perfect unless the adversary asks $(\tilde{x}, a)$ as a query for $\mathcal{H}_1$. Note that, since the adversary is active, the adversary cannot change or corrupt the secret long-term key of the peer. This time the value of σ is similar to a signature forgery and we can apply the Forking Lemma [3,39] to extract the CDH of U and V.

Proof Recall that cNR-eCK security means security in an ASICS model that omits the session-key query, so the allowed queries are $\mathcal{Q}_N \cup \{\text{corrupt}, \text{randomness}\}$. The freshness condition remains unchanged.

Recall further that the goal of the adversary is to *recovery* the session string; let S be the event that an algorithm $\mathcal{M}$ computes the session string. The security proof largely follows the original proof of Ustaoglu that CMQV is eCK-secure, but can be simplified somewhat as the queries in the cNR-eCK game are restricted compared to full eCK security.

Consider the following two complementary events:

- E. There exists a certificate C' (created using hregister) such that $\mathcal{M}$, during its execution, queries $\mathcal{H}_1(*, b)$ (where $C'.\text{pk} = g^b$) before issuing any corrupt$(C'.\text{pk})$ query (if it issues one at all).
- $\overline{E}$. During its execution, for every certificate C' (created using hregister) for which $\mathcal{M}$ queries $\mathcal{H}_1(*, b)$ (where $C'.\text{pk} = g^b$), it issued a corrupt$(C'.\text{pk})$ query before the $\mathcal{H}_1(*, b)$ query.

Since the events are complementary, if $\mathcal{M}$ succeeds in computing the session string, it succeeded either when E occurred or when $\overline{E}$ occurred.

We will see how, when each event occurs, the required polynomial-time reduction exists.

Event E Suppose event E occurs and $\mathcal{M}$ succeeds in computing the session string.

Here, the simulator $\mathcal{S}$ guesses one public key pk^* at random and assigns $\text{pk}^* \leftarrow V$, where (U, V) is the Diffie–Hellman challenge. All other public keys are generated according to the protocol specification.

For all sessions and queries where the session actor is not using pk^*, $\mathcal{S}$ follows the protocol specification exactly.

For sessions where the session actor is using pk^*, $\mathcal{S}$ responds to queries as follows:

- hregister$(\text{pk}^*, \hat{P})$: $\mathcal{S}$ outputs a certificate as normal.
- corrupt(pk^*): $\mathcal{S}$ aborts.
- randomness$(s = (C, i))$ where $C.\text{pk} = \text{pk}^*$: Return s_{rand}.

For sessions where the session actor is using pk^* and is the initiator, $\mathcal{S}$ responds to queries as follows:

- create$(s = (C, i), \mathcal{I}, C')$ where $C.\text{pk} = \text{pk}^*$: $\mathcal{S}$ selects $x \in_R \mathbb{Z}_q$, computes $X \leftarrow g^x$, and responds with X. Note that s_{rand} is not used in the calculation.
- send$(s = (C, i), M)$ where $s_{\text{acert}}.\text{pk} = \text{pk}^*$ and $s_{\text{role}} = \mathcal{I}$: $\mathcal{S}$ does not need to simulate anything here, since there is not outgoing message required, and since the only variable updated is the session string ss but no session-key reveal query is allowed.

For sessions where the session actor is using pk^* and is the responder, $\mathcal{S}$ responds to queries as follows:

- create$(s = (C, i), \mathcal{R}, C')$ where $C.\text{pk} = \text{pk}^*$: no response required.

– $\mathsf{send}(s = (C, i), M)$ where $s_{\text{acert}}.\mathsf{pk} = \mathsf{pk}^*$ and $s_{\text{role}} = \mathcal{R}$: $\mathcal{S}$ selects $y \in_R \mathbb{Z}_q$, computes $Y \leftarrow g^y$, and responds with Y. Note that s_{rand} is not used in the calculation.

$\mathcal{S}$ responds to $\mathcal{H}_2$ queries as normal. $\mathcal{S}$ responds to $\mathcal{H}_1(*, b)$ queries as normal for all b such that $g^b \neq \mathsf{pk}^*$. When $\mathcal{M}$ queries $\mathcal{H}_1(*, b)$ where $g^b = \mathsf{pk}^* = V$, $\mathcal{S}$ outputs the solution to the Diffie–Hellman challenge (U, V) as U^b.

Note that $\mathcal{S}$'s simulation is perfect up until an abort event from the $\mathsf{corrupt}$ query occurs. Given that event E occurs, there exists some public key $\mathsf{pk} = g^b$ for which the query $\mathcal{H}_1(*, b)$ occurs before any $\mathsf{corrupt}(\mathsf{pk})$ query occurs. With probability at least $1/n_{\mathsf{kgen}}$, where n_{kgen} is the number of kgen queries made by $\mathcal{M}$, $\mathcal{S}$ this condition holds for pk^*. When $\mathcal{S}$ guesses correctly, $\mathcal{M}$ will indeed query $\mathcal{H}_1(*, b)$ before any $\mathsf{corrupt}(\mathsf{pk}^*)$ query and thus $\mathcal{S}$ will solve the computational Diffie–Hellman problem.

Thus, when event E occurs, there exists a polynomial-time reduction from a $\mathsf{cNR\text{-}eCK}$ adversary for the session string variant of CMQV to the computational Diffie–Hellman problem under the assumption that $\mathcal{H}_1$ is a random oracle, with a tightness factor of n_{kgen}.

Event $\overline{E}$ We divide this event into two complementary cases:

– M. The session s for which the adversary output the session string has an M2-matching session s'.
– $\overline{M}$. The session s for which the adversary output the session string does not have an M2-matching session.

When $\overline{E}$ occurs, either M or $\overline{M}$ must also occur.

Event $\overline{E} \wedge M$ Suppose event $\overline{E}$ occurs and there is an M2-matching session s' for the target session s.

Here, the simulator guesses two sessions s and s'; assume without loss of generality that $s_{\text{role}} = \mathcal{I}$ and $s'_{\text{role}} = \mathcal{R}$. $\mathcal{S}$ responds to all kgen, $\mathsf{hregister}$, $\mathsf{corrupt}$, and $\mathsf{randomness}$ queries as specified by the protocol. For all sessions other than s and s', $\mathcal{S}$ responds to create and send as specified by the protocol. For s and s', $\mathcal{S}$ responds to create and send as follows:

– $\mathsf{create}(s = (C, i), \mathcal{I}, C')$: Return $X \leftarrow U$, where (U, V) is the Diffie–Hellman challenge. Note that s_{rand} is not used.
– $\mathsf{create}(s' = (C', i), \mathcal{R}, C)$: No response required.
– $\mathsf{send}(s', M)$: Return $Y \leftarrow V$, where (U, V) is the Diffie–Hellman challenge. Note that s'_{rand} is not used.

$\mathcal{S}$ responds to $\mathcal{H}_2$ queries as normal. $\mathcal{S}$ responds to $\mathcal{H}_1$ queries as normal except for the queries $(\tilde{x}, a)$ or $(\tilde{y}, b)$, where a and b are the secret keys corresponding to the public keys in sessions s and s'; when this occurs, the simulation aborts.

Note that $\mathcal{S}$'s simulation is perfect unless a $\mathcal{H}_1(\tilde{x}, a)$ or $\mathcal{H}_1(\tilde{y}, b)$ query occurs. Because of event $\overline{E}$, $\mathcal{M}$ issues a $\mathsf{corrupt}(g^a)$ query before any $\mathcal{H}_1(\tilde{x}, a)$ query, and a $\mathsf{corrupt}(g^b)$ query before any $\mathcal{H}_1(\tilde{y}, b)$ query. Since $\tilde{x}$ and $\tilde{y}$ are used in only one session and $\mathcal{H}_1$ is a random function, no information can be learned about $\tilde{x}$ and $\tilde{y}$ without $\mathsf{randomness}(s)$ or $\mathsf{randomness}(s')$ queries. By the freshness condition, it cannot be that both $\mathsf{randomness}(s)$ and $\mathsf{corrupt}(g^a)$ occurred, or that both $\mathsf{randomness}(s')$ and $\mathsf{corrupt}(g^b)$ occurred. Thus, if $\mathcal{S}$ correctly guess s and s', the simulation is perfect and does not abort. This happens with probability at least $2/n^2_{\mathsf{create}}$.

Assuming the simulation is perfect and does not abort and that $\mathcal{M}$ outputs the session string, $\mathcal{S}$ can use this to solve the Diffie–Hellman problem. In particular, let σ be the shared secret in the session string output by $\mathcal{M}$. Then, $\mathcal{S}$ outputs $\sigma g^{-abed} U^{-be} V^{-ad}$ as the solution to the computational Diffie–Hellman challenge (U, V).

Thus, when event $\overline{E} \wedge M$ occurs, there exists a polynomial-time reduction from a $\mathsf{cNR\text{-}eCK}$ adversary for the session string variant of CMQV to the computational Diffie–Hellman problem under the assumption that $\mathcal{H}_1$ is a random oracle, with a tightness factor of n^2_{create}.

Event $\overline{E} \wedge \overline{M}$ Suppose event $\overline{E}$ occurs but there is no M2-matching session for the target session s.

Here, the simulator guesses integers $j \in_R \{1, \ldots, n_{\mathsf{kgen}}\}$, and a session s^*. Assume without loss of generality that $s^*_{\text{role}} = \mathcal{I}$.

For the jth query to kgen, $\mathcal{S}$ assigns $\mathsf{pk}^* \leftarrow V$ from the Diffie–Hellman challenge (U, V) to be the public key; for all other kgen queries, it responds as specified by the protocol.

All $\mathsf{hregister}$ queries are responded to as normal. All $\mathsf{corrupt}$ queries are responded to as normal, except for $\mathsf{corrupt}(\mathsf{pk}^*)$, in which case $\mathcal{S}$ aborts.

Suppose that $\mathcal{M}$ selects s^* as the target session and furthermore that $s^*_{\text{pcert}}.\mathsf{pk} = V$.

For all sessions and queries where the session actor or peer is not using pk^*, $\mathcal{S}$ follows the protocol specification exactly.

For sessions where the session actor is using pk^*, $\mathcal{S}$ responds as in event E.

For sessions where the session peer is using pk^*, $\mathcal{S}$ responds as specified by the protocol, except for the target session s^*. In s^*, $\mathcal{S}$ responds as follows:

– $\mathsf{create}(s^*, \mathcal{I}, C')$: $\mathcal{S}$ returns $X \leftarrow U$, where (U, V) is the Diffie–Hellman challenge. Note that s^*_{rand} is not used.
– $\mathsf{send}(s^*, M)$: No response required.
– $\mathsf{randomness}(s^*)$: Return s^*_{rand}.
– $\mathsf{session\text{-}key}(s^*)$: $\mathcal{S}$ aborts. Assuming that $\mathcal{S}$ correctly guesses s^* as the target session, this abort will never occur.

$\mathcal{S}$ responds to $\mathcal{H}_2$ queries as normal. $\mathcal{S}$ responds to $\mathcal{H}_1(\tilde{x}, b)$ queries as normal except for the following two cases:

- If $g^b = \mathsf{pk}^*$: $\mathcal{S}$ outputs the solution to the Diffie–Hellman challenge (U, V) as U^b.
- If $\tilde{x} = s^*_{\text{rand}}$ and $g^a = s^*_{\text{acert}}.\mathsf{pk}$: $\mathcal{S}$ aborts.

Note that $\mathcal{S}$'s simulation is perfect up until an abort event from the $\mathsf{corrupt}$ or the $\mathcal{H}_2$ query occurs. Given that s^* is fresh and no matching session exists, no $\mathsf{corrupt}(s^*)$ query is allowed and hence $\mathcal{S}$ does not abort for that reason. Given that event $\overline{E}$ occurs, if $\mathcal{M}$ queries $\mathcal{H}_1(s^*_{\text{rand}}, a)$ such that $g^a = s^*_{\text{acert}}.\mathsf{pk}$, $\mathcal{M}$ must have issued a $\mathsf{corrupt}(g^a)$ query first. But it is also the case that s^* is fresh, so $\mathcal{M}$ cannot have also issued a $\mathsf{randomness}(s^*)$ query, and thus cannot know s^*_{rand} unless it guessed it correctly, which can be done only with negligible probability.

Assume the simulation is perfect and does not abort, and that $\mathcal{M}$ outputs the session string containing the correct shared secret $\sigma = g^{uy} g^{ady} g^{uve} g^{adev}$. $\mathcal{S}$ can then compute $\eta = \sigma Y^{-ad} V^{-ade} = g^{uy+uve}$. But the peer's ephemeral secret key y was chosen by the adversary, so without y $\mathcal{S}$ cannot directly compute g^{uv} from η.

Using the Forking Lemma, $\mathcal{S}$ runs $\mathcal{M}$ on the same input and the same random coins but with modified answers to $\mathcal{H}_2$ queries. Note that $\mathcal{M}$ must have queried $\mathcal{H}_2(Y, s^*_{\text{acert}}.\mathsf{id}, s^*_{\text{pcert}}.\mathsf{id})$ to obtain e, because otherwise $\mathcal{M}$ would be unable to compute σ except with negligible probability. For the second run of $\mathcal{M}, \mathcal{S}$ responds to $\mathcal{H}_2(Y, s^*_{\text{acert}}.\mathsf{id}, s^*_{\text{pcert}}.\mathsf{id})$ with $e' \neq e$ selected uniformly at random.

If $\mathcal{M}$ succeeds in the second run, it outputs

$$\sigma = g^{uy} g^{ad'y} g^{uve'} g^{ad'e'v}.$$

$\mathcal{S}$ can then compute

$$\eta' = \sigma' Y^{-ad'} V^{-ad'e'} = g^{uy} g^{uve'}.$$

$\mathcal{S}$ can furthermore compute

$$\begin{aligned}(\eta/\eta')^{1/(e-e')} &= (g^{uy} g^{uve} g^{-uy} g^{-uve'})^{1/(e-e')} \\ &= g^{uv(e-e')/(e-e')} = g^{uv}\end{aligned}$$

which is the solution to the computational Diffie–Hellman challenge (U, V).

Thus, when event $\overline{E} \wedge \overline{M}$ occurs, there exists a polynomial-time reduction from a $\mathsf{cNR\text{-}eCK}$ adversary for the session string variant of CMQV to the computational Diffie–Hellman problem under the assumption that $\mathcal{H}_1$ and $\mathcal{H}_2$ are random oracles, with a tightness factor of $n_{\mathsf{create}} n_{\mathsf{kgen}} n_{\mathcal{H}_2} c$, where c is a constant from the Forking Lemma. □

Remark 12 Because in the above lemma we do not have to prove full session key indistinguishability security of CMQV, instead proving the hardness of session string computation of a variant of CMQV, we can make a few simplifications from Ustaoglu's original proof:

- We do not have to worry about key replication attacks (when the adversary causes two non-matching sessions to have the same session key (that is, session string), and then reveals the session key at one of the sessions) because there is no $\mathsf{session\text{-}key}$ query.
- In event E, we do not have to worry about setting the session string correctly for any session involving the user whose public key has been injected with the CDH challenge, because there is no $\mathsf{session\text{-}key}$ query. Thus we do not need a DDH oracle here.
- In event $\overline{E} \wedge M$, we do not have to use the DDH oracle to test which of the many $\mathcal{H}$ random oracle queries is the solution we need: we simply output the CDH value derived directly from the output of $\mathcal{M}$.

Lemma 3 *The session string decision problem for CMQV is poly-time reducible to the decisional problem of the Diffie–Hellman relation ϕ.*

Proof Let D be a polynomial-time algorithm that can distinguish real CMQV session strings $(g^{(y+eb)(x+da)} \parallel X \parallel Y \parallel \mathsf{id} \parallel A \parallel \mathsf{id}' \parallel B)$ from random session strings $(g^r \parallel g^x \parallel g^y \parallel \mathsf{id} \parallel g^a \parallel \mathsf{id}' \parallel g^b)$, for randomly chosen $a, b, x, y, r \in_R \mathbb{Z}_q$, id and id' are arbitrary binary strings, $d = \mathcal{H}_2(g^x \parallel \mathsf{id} \parallel \mathsf{id}')$, and $e = \mathcal{H}_2(g^y \parallel \mathsf{id} \parallel \mathsf{id}')$.

We claim that there exists an algorithm E that can distinguish real Diffie–Hellman triples (g^u, g^v, g^{uv}) from random triples (g^u, g^v, g^w) for randomly chosen $u, v, w \in_R \mathbb{Z}_q$.

First, note that $g^{(y+eb)(x+da)} = g^{xy+ady+bex+abde}$. Using D construct E_D as follows. Let (U, V, W) be a Diffie–Hellman challenge. Pick arbitrary $\mathsf{id}, \mathsf{id}'$. Do one of the following, each with equal probability:

1. Set $A \leftarrow U$ and $B \leftarrow V$. Choose $x, y \in_R \mathbb{Z}_q$.
   Run D on the session string $(g^{xy} A^{dy} B^{ex} W^{de} \parallel g^x \parallel g^y \parallel \mathsf{id} \parallel A \parallel \mathsf{id}' \parallel B)$.
2. Set $A \leftarrow U$ and $Y \leftarrow V$. Choose $x, b \in_R \mathbb{Z}_q$.
   Run D on the session string $(Y^x W^d g^{bex} A^{bde} \parallel g^x \parallel Y \parallel \mathsf{id} \parallel A \parallel \mathsf{id}' \parallel g^b)$.
3. Set $X \leftarrow U$ and $B \leftarrow V$. Choose $a, y \in_R \mathbb{Z}_q$.
   Run D on the session string $(X^y g^{ady} W^e B^{ade} \parallel X \parallel g^y \parallel \mathsf{id} \parallel g^a \parallel \mathsf{id}' \parallel B)$.
4. Set $X \leftarrow U$ and $Y \leftarrow V$. Choose $a, b \in_R \mathbb{Z}_q$.
   Run D on the session string $(W Y^{ad} X^{be} g^{abde} \parallel X \parallel Y \parallel \mathsf{id} \parallel g^a \parallel \mathsf{id}' \parallel g^b)$.

E outputs the result of D.

Note that in each of the above cases, if (U, V, W) is a real Diffie–Hellman triple, then D is run on a real CMQV session string, whereas if (U, V, W) is a random triple, then D is run on a random session string. Thus, if D is a distinguisher for CMQV session strings, then E is a distinguisher for the Diffie–Hellman relation. □

References

1. Adams, C., Farrell, S., Kause, T., Mononen, T.: Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP). RFC 4210 (Proposed Standard). http://www.ietf.org/rfc/rfc4210.txt, updated by RFC 6712 (2005)
2. Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: Recommendation for key management—part 1: general. NIST Special Publication. http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf (2007)
3. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Juels, A., Wright, R.N., Vimercati, S. (eds.) ACM CCS 06, pp. 390–399. ACM Press, Alexandria (2006)
4. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
5. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
6. Bellare, M., Rogaway, P.: Provably secure session key distribution: the three party case. In: 27th ACM STOC, pp. 57–66. ACM Press, Las Vegas (1995)
7. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. In: Darnell, M. (ed.) 6th IMA International Conference on Cryptography and Coding. LNCS, vol. 1355, pp. 30–45. Springer, Heidelberg (1997)
8. Blake-Wilson, S., Menezes, A.: Entity authentication and authenticated key transport protocols employing asymmetric techniques. In: Christianson, B., Crispo, B., Lomas, T.M.A., Roe, M. (eds.) Proceedings of the 5th International Workshop on Security Protocols, Paris, France, April 7–9, 1997. LNCS, vol. 1361, pp. 137–158. Springer (1998)
9. Blake-Wilson, S., Menezes, A.: Unknown key-share attacks on the station-to-station (STS) protocol. In: Imai, H., Zheng, Y. (eds.) PKC'99. LNCS, vol. 1560, pp. 154–170. Springer, Heidelberg (1999)
10. Boldyreva, A., Fischlin, M., Palacio, A., Warinschi, B.: A closer look at PKI: security and efficiency. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 458–475. Springer, Heidelberg (2007)
11. Boyd, C., Cremers, C., Feltz, M., Paterson, K.G., Poettering, B., Stebila, D.: ASICS: Authenticated key exchange security incorporating certification systems. In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 381–399. Springer, Heidelberg (2013)
12. CA/Browser Forum: Baseline requirements for the issuance and management of publicly-trusted certificates, v1.1.6. https://cabforum.org/wp-content/uploads/Baseline_Requirements_V1_1_6.pdf (2013)
13. CA/Browser Forum: Guidelines for the issuance and management of extended validation certificates, v1.4.3. https://cabforum.org/wp-content/uploads/Guidelines_v1_4_3.pdf (2013)
14. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg, Germany (2001)
15. Cash, D., Kiltz, E., Shoup, V.: The twin Diffie–Hellman problem and applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008)
16. Chatterjee, S., Menezes, A., Ustaoglu, B.: Combined security analysis of the one- and three-pass unified model key agreement protocols. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 49–68. Springer, Heidelberg (2010)
17. Choo, K.K.R., Boyd, C., Hitchcock, Y.: Examining indistinguishability-based proof models for key establishment protocols. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 585–604. Springer, Heidelberg (2005)
18. Cremers, C.: Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In: Cheung, B.S.N., Hui, L.C.K., Sandhu, R.S., Wong, D.S. (eds.) ASIACCS 11, pp. 80–91. ACM Press, Hong Kong, China (2011)
19. Cremers, C.J.F., Feltz, M.: Beyond eCK: perfect forward secrecy under actor compromise and ephemeral-key reveal. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 734–751. Springer, Heidelberg (2012)
20. Cremers, C.J.F., Feltz, M.: Beyond eCK: perfect forward secrecy under actor compromise and ephemeral-key reveal. Des. Codes Cryptogr. **74**(1), 183–218 (2015)
21. Ducklin, P.: The TURKTRUST SSL certificate fiasco—what really happened, and what happens next? http://nakedsecurity.sophos.com/2013/01/08/the-turktrust-ssl-certificate-fiasco-what-happened-and-what-happens-next/ (2013)
22. Farshim, P., Warinschi, B.: Certified encryption revisited. In: Preneel, B. (ed.) AFRICACRYPT 09. LNCS, vol. 5580, pp. 179–197. Springer, Heidelberg (2009)
23. FOX IT: Black Tulip: Report of the investigation into the DigiNotar Certificate Authority breach. http://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/rapporten/2012/08/13/black-tulip-update/black-tulip-update.pdf (2012)
24. Freire, E.S.V., Hofheinz, D., Kiltz, E., Paterson, K.G.: Non-interactive key exchange. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 254–271. Springer, Heidelberg (2013)
25. Goldberg, I., Stebila, D., Ustaoglu, B.: Anonymity and one-way authentication in key exchange protocols. Des. Codes Cryptogr. **67**(2), 245–269 (2013)
26. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984)
27. Jeong, I.R., Katz, J., Lee, D.H.: One-round protocols for two-party authenticated key exchange. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 04. LNCS, vol. 3089, pp. 220–232. Springer, Heidelberg (2004)
28. Kaliski, B.S.: An unknown key-share attack on the MQV key agreement protocol. ACM Trans. Inf. Syst. Secur. **4**, 275–288 (2001)
29. Krawczyk, H.: HMQV: A high-performance secure Diffie–Hellman protocol. Cryptology ePrint Archive, Report 2005/176. http://eprint.iacr.org/2005/176 (2005)
30. Krawczyk, H.: HMQV: A high-performance secure Diffie–Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
31. Kudla, C., Paterson, K.G.: Modular security proofs for key agreement protocols. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 549–565. Springer, Heidelberg (2005)
32. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)

33. Lauter, K., Mityagin, A.: Security analysis of KEA authenticated key exchange protocol. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 378–394. Springer, Heidelberg (2006)
34. Lim, C.H., Lee, P.J.: A key recovery attack on discrete log-based schemes using a prime order subgroup. In: Kaliski Jr., B.S. (ed.) CRYPTO'97. LNCS, vol. 1294, pp. 249–263. Springer, Heidelberg (1997)
35. McCurley, K.S.: A key distribution system equivalent to factoring. J Cryptol **1**(2), 95–105 (1988)
36. Menezes, A.: Another look at HMQV. Cryptology ePrint Archive, Report 2005/205. http://eprint.iacr.org/2005/205 (2005)
37. Menezes, A., Ustaoglu, B.: On the importance of public-key validation in the MQV and HMQV key agreement protocols. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 133–147. Springer, Heidelberg (2006)
38. Menezes, A., Ustaoglu, B.: Security arguments for the UM key agreement protocol in the NIST SP 800–56A standard. In: Abe, M., Gligor, V. (eds.) ASIACCS 08. pp. 261–270. ACM Press, Tokyo, Japan (2008)
39. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J Cryptol **13**(3), 361–396 (2000)
40. Ristenpart, T., Yilek, S.: The power of proofs-of-possession: securing multiparty signatures against rogue-key attacks. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 228–245. Springer, Heidelberg (2007)
41. Schaad, J.: Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF). RFC 4211 (Proposed Standard). http://www.ietf.org/rfc/rfc4211.txt (2005)
42. Shmuely, Z.: Composite Diffie–Hellman public-key generating systems are hard to break. Technical Report No. 356, Computer Science Department, Technion-Israel Institute of Technology (1985)
43. Shoup, V.: On formal methods for secure key exchange (version 4) (November 1999), revision of IBM Research Report RZ 3120. http://www.shoup.net/papers/skey.pdf (1999)
44. Turner, S.: The application/pkcs10 Media Type. RFC 5967 (Informational). http://www.ietf.org/rfc/rfc5967.txt (2010)
45. Turner, P., Polk, W., Barker, E.: ITL Bulletin for July 2012: preparing for and responding to certification authority compromise and fraudulent certificate issuance. http://csrc.nist.gov/publications/nistbul/july-2012_itl-bulletin.pdf (2012)
46. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Des. Codes Cryptogr. **46**(3), 329–342 (2008)
47. Ustaoglu, B.: Comparing sessionstatereveal and ephemeralkeyreveal for Diffie–Hellman protocols. In: Pieprzyk, J., Zhang, F. (eds.) ProvSec 2009. LNCS, vol. 5848, pp. 183–197. Springer, Heidelberg (2009)