A Modern View on Forward Security[*]

Colin Boyd[1] and Kai Gellert[2]

[1]NTNU, Norwegian University of Science and Technology, Trondheim, Norway,
colin.boyd@ntnu.no
[2]University of Wuppertal, Wuppertal, Germany, kai.gellert@uni-wuppertal.de

Abstract

Forward security ensures that compromise of entities today does not impact the security of cryptographic primitives employed in the past. Such a form of security is regarded as increasingly important in the modern world due to the existence of adversaries with mass storage capabilities and powerful infiltration abilities. Although the idea of forward security has been known for over 30 years, current understanding of what it really should mean is limited due to the prevalence of new techniques and inconsistent terminology. We survey existing methods for achieving forward security for different cryptographic primitives and propose new definitions and terminology aimed at a unified treatment of the notion.

1 Introduction

Providing confidentiality to sensitive data is a standard property of encryption schemes. Without knowledge of the decryption key, an observing adversary should not be able to decrypt any encrypted traffic. Nowadays, in the presence of powerful nation-state adversaries that establish Internet surveillance programs, achieving the necessary security for sensitive data has become more challenging. Even if Internet traffic is encrypted, a resourceful adversary could collect encrypted traffic on a massive scale and store it in dedicated data centres. Should a decryption key be compromised at a later point in time, for example by database breaches or by application of legislative means in various jurisdictions, the respective encrypted traffic could be decrypted again. It is therefore evident that encryption protocols should take into account the threat from adversaries who may practise mass-storage of encrypted data.

In order to prevent such attacks on cryptographic schemes, some key material that was used to decrypt traffic originally must no longer be available when key compromise occurs at a later time. One option is to delete or update the decryption key directly so that it cannot be recovered or rolled back to an earlier version. This is the basis for what is usually known as *forward-secure* encryption and achieves *forward security.* Another option is to make use of a session key which is used to protect data, is deleted after use, and cannot be recovered using other stored keys. This second option is most often known as *forward secrecy* in the context of key exchange protocols. Later in this paper we will compare the different terminology and possible alternative meanings of forward security and forward secrecy.

[*] Supported by the Research Council of Norway under Project No. 248166 and the German Research Foundation (DFG) under project JA 2445/2-1. An extended abstract of this work appeared in the Computer Journal. This is the full version.

Forward Secrecy in Key Exchange. To illustrate the increasing importance of forward secrecy in key exchange, it is useful to look at the case of the Transport Layer Security (TLS) protocol (still often referred to as SSL for historical reasons). TLS is the most widely used security protocol on the Internet and has passed through many versions. The newest version is TLS 1.3 [47] which *only* allows key exchange with forward secrecy. Earlier versions, some of which are still almost universally deployed and supported, allow two kinds of key exchange: a version using RSA encryption [48] which does not provide forward secrecy, and a version using Diffie–Hellman key exchange [21] which does provide forward secrecy. As recently as 2014[1] less than 50% of prominent web servers even supported the option of providing forward secrecy, but by 2019 this number had increased to more than 97% and most servers select forward secrecy by default today.

Forward Secrecy without Interaction. Typical key exchange protocols require interaction between the participants. This interaction plays a critical role in achieving forward secrecy. Non-interactive protocols, in which secured data leaves the sender before any response from the recipient, cannot apply the same techniques. Before going further we should emphasise that there are different possible meanings of the term *non-interactive* used in the literature. In particular, protocols for non-interactive key exchange (NIKE) [25] require that *no* messages flow between the parties. In contrast, in this paper we use the term non-interactive to mean that messages do not flow in both directions, thereby allowing protocols where one of the two parties is offline during protocol execution. Some authors use the term *asynchronous* where we use *non-interactive.*

Prominent examples of non-interactive protocols (in our sense) include electronic mail and instant messaging, where the recipient may be offline during communication. For the case of email, standardised end-to-end security solutions such as GPG [35] or S-MIME [45, 46], do not provide forward secrecy. In contrast, there has been much effort in providing high security for instant messaging; protocols such as Signal [51] do emphasise the need for forward secrecy even for the initial messages when no interaction has yet occurred. In recent years, interest in non-interactive protocols has increased even in Internet services where interaction could be used, due to the desire to achieve higher efficiency and reduce delay. This interest is exemplified by several Internet companies developing and experimenting with protocols that allow clients to send encrypted payload data before obtaining any fresh input from the server, for example Google's QUIC protocol [12], Facebook's Zero protocol [32], and the TLS 1.3 0-RTT mode [47].

A Confusing Landscape. Due to the lack of interactivity, traditional techniques to achieve forward secrecy (elaborated in detail in the next section) cannot be used in the protocols mentioned above. Therefore alternative techniques have been designed, and in some cases deployed, which do not fit the traditional view. This has led to considerable confusion about what forward secrecy should mean, whether there are different kinds of forward secrecy, and whether forward secrecy in key exchange is different from the meaning of forward *security* for key exchange or other primitives. Due to the different possible meanings of forward secrecy, it is possible to reach contradictory conclusions on what is possible to achieve with non-interactive protocols. Some authors [58, 57, 10, 11, 34] state that it is impossible to achieve forward secrecy without interaction while others [29, 3, 37, 56] state that fully equivalent forward secrecy has been achieved in their non-interactive constructions.

The lack of consistent terminology is another symptom of the confusion existing in the area. There are very many different adjectives (perfect, partial, weak, and so on) which have been

[1] Figures taken from the Qualys SSL Labs web page: `https://www.ssllabs.com/ssl-pulse/`

used to qualify different flavours of forward secrecy and forward security. There are instances of different terminology used to describe the same thing, and of the same terminology used to describe different things. In Appendix A we collect many of these terms, explain their origins, and compare their meanings.

Our Contributions. The goal of this paper is to shed some light on competing definitions for forward secrecy and forward security, and to propose a unified view on how to compare both non-interactive and interactive protocols with forward security.

- We discuss and compare the different terminology and possible alternative meanings of forward security/secrecy leading us to the conclusion to view forward security as a generalisation of forward secrecy in key exchange (Sections 2, 3).
- We explain different techniques to achieve forward secrecy in non-interactive key exchange and identify message suppression as an attack vector to bypass forward secrecy under certain circumstances (Section 4).
- We identify different properties of forward security based on a timing parameter, differentiate the different key types that can be used to achieve these properties, and show how known types of forward-secure primitives map to the different key types. This allows a meaningful comparison of the degree of forward security which is achieved by different cryptographic schemes, even when they use quite different techniques (Section 5).

2 The Traditional View on Forward Secrecy

During the 1970s the academic view of authenticated key exchange (AKE) was initially established. A seminal paper of Needham and Schroeder [41] categorised AKE protocols as either public-key based or symmetric-key based. In either case, protocol parties each possess a *long-term key* which is chosen when a user is enrolled into the system and the key remains unchanged during the system lifetime.[2] In the symmetric-key setting each user typically shares a long-term key with a trusted server. For example, in a corporate setting an employee may be issued a long-term key when first joining the company. In the public-key setting a long-term private key is generated by, or on behalf of, each user and the corresponding public key may typically be published together with a valid certificate. In this paper we focus mainly on public-key based protocols, but will not discuss further the problem of certifying, or otherwise managing, public keys.

The goal of an AKE protocol is to establish a new *session key* to be used to protect communications in a subsequent data exchange phase. Generally the security goal is achieved if no efficient adversary can reliably distinguish a new session key from a completely random string chosen from the key space. It is assumed that the adversary is able to eavesdrop on all messages sent between protocol parties and also to alter messages or fabricate new ones using any available information. A general principle is that different session keys must provide independent protection so that compromise of one session key does not lead to compromise of any other session key. In order to model this property we assume that the adversary has the ability to obtain session keys from any protocol run other than the one under attack.

It is evident from these requirements that session keys must be generated in some way that relies on randomised input from at least one of the protocol parties. Such randomness may be used directly to specify a session key or it may be used as an input to other functions as in the

[2] In practice even long-term keys have a lifetime but for the sake of simplicity we ignore that.

Diffie–Hellman protocol described later in this section. In either case such random key material is regarded as *ephemeral* because it is used only while the protocol is being run and deleted when the protocol is complete. An adversary in possession of some party's complete state, particularly both long-term and ephemeral keys, is able to compute any session keys since the adversary also sees all messages received by the party. How can we then provide any security for devices which may be compromised? It turns out that in some cases we can retain security of session keys if compromise happens after the session is finished. This is the idea behind forward secrecy.

2.1 A Simple Key Transport Protocol

Consider first a simple protocol in the symmetric-key setting shown in Figure 1, run between two parties A and B. Initially A and B already share a long-term private key K_{AB} but, in order to isolate different sessions, they use the protocol to set up a new session key k which is chosen by party A and sent to party B; this technique is often known as *key transport.* The key k is the ephemeral key generated by A in this protocol, and k will also be the session key. It is sent to party B encrypted with the long-term key K_{AB}. The field N_B sent in message 1 is a random value chosen by B and checked by B on receipt of message 2. This allows B to verify that the session key is new, since message 2 cannot be replayed from an earlier run of the protocol.

1. $B \to A: \quad N_B$
2. $A \to B: \quad \{N_B, B, k\}_{K_{AB}}$

Figure 1: ISO/IEC 11770-2 Key Establishment Mechanism 4 [31]. A protocol without forward secrecy.

Suppose that an adversary records a protocol run and later wants to find the session key k (and therefore recover all of the traffic protected by k). If the adversary is able to compromise either A or B, thereby obtaining the long-term key K_{AB}, then it is trivial for the adversary to decrypt the recorded message 2 and obtain the session key k.

2.2 Diffie–Hellman Key Agreement

The key agreement protocol of Diffie and Hellman [21], illustrated in Figure 2, is one of oldest constructions in public-key cryptography. Two parties A and B agree publicly on an element g that generates a multiplicative group G. They then select random values, r_A and r_B respectively, in the range between 1 and the order of G. A calculates $t_A = g^{r_A}$ and B calculates $t_B = g^{r_B}$ and they exchange these values. A shared secret, $Z_{AB} = g^{r_A r_B}$ can be calculated by both A and B due to the homomorphic property of exponentiation: $Z_{AB} = t_A^{r_B} = t_B^{r_A}$. Finally, a key derivation function, KDF, will be applied to Z_{AB} and other public inputs to obtain the session key: $k = KDF(Z_{AB}, \ldots)$.

The basic Diffie–Hellman protocol shown in Fig. 2 lacks authentication and is therefore insecure against active adversaries – neither A nor B knows which party the secret Z_{AB} is shared with. There are different ways that long-term keys can be applied to authenticate the Diffie–Hellman messages, for example by using digital signatures, but we avoid giving an explicit choice here. The ephemeral secrets are the values r_A and r_B which will be deleted once the session is completed. This means that an adversary who later compromises A or B (or both) will only obtain the long-term keys used to authenticate, which are independent of the session key. Thus the adversary will only have the ephemeral public keys, g^{r_A} and g^{r_B}, exchanged during the protocol but needs to find $g^{r_A r_B}$ in order to learn the shared secret used to form k.

$$
\begin{array}{ccc}
A & & B \\
r_A \xleftarrow{\$} \mathbb{Z}_q & & \\
t_A = g^{r_A} & & \\
 & \xrightarrow{\quad t_A \quad} & r_B \xleftarrow{\$} \mathbb{Z}_q \\
 & & t_B = g^{r_B} \\
Z_{AB} = t_B^{r_A} & \xleftarrow{\quad t_B \quad} & Z_{AB} = t_A^{r_B}
\end{array}
$$

Figure 2: Diffie–Hellman key agreement. A protocol with forward secrecy.

This is the so-called Diffie–Hellman problem which is believed to be computationally intractable for a well-chosen group G (at least until quantum computers become practical [50]).

2.3 Defining Forward Secrecy

We have seen that some protocols can be vulnerable to adversary compromise after sessions are complete, while others remain secure. Those that remain secure in such a situation are said to possess the property of *forward secrecy.* The concept and terminology were introduced into the academic literature by Günther in 1989 [28].

Definition 1. An AKE protocol provides *forward secrecy* if compromise of long-term secrets does not lead to compromise of session keys of previously completed sessions.

The Diffie–Hellman protocol has been used as the basis of hundreds of key exchange protocols and is often considered as the fundamental basis for forward secrecy. It is widely deployed in real-world protocols such as TLS, IPSec and SSH. However, we take the opportunity to note that *any* public-key encryption scheme can be adapted to provide forward secrecy [43]. To do this, party A first chooses an ephemeral public/private key pair and sends the public encryption key to party B. B then chooses a session key and returns it to A encrypted with the ephemeral public key. After the session is completed the ephemeral private key is deleted by A and so will not become available to a later adversary who compromises A.

Forward secrecy was originally defined by Günther [28] in the context of *identity-based cryptography.* In this context there is a master long-term key belonging to a trusted authority (TA) which is used to generate the long-term keys of protocol entities. Here we may differentiate between compromise of individual entity long-term keys and compromise of the TA master key, which implicitly compromises all entities' long-term keys. Some protocols are secure against the former type of compromise but not the latter type [15].

We conclude this section by observing that *interaction* seems to play a fundamental role in obtaining the forward secrecy property. Although the protocol of Figure 1 shows that interaction is certainly not sufficient to provide forward secrecy, it may be difficult to see how forward secrecy can be obtained without interaction. If a party A does not provide an ephemeral input to the protocol then the adversary will later get all the secrets used by A and so should be able to compute the session key in the same way as A. We will see later how this apparent impossibility result can be overcome in non-interactive protocols.

3 Forward Security as Generalization

Forward secrecy for key exchange is concerned with ensuring that previously used session keys are not revealed following compromise of participants at some later time. The name itself carries an implication that forward secrecy is a form of data confidentiality, with the purpose to prevent leakage of user data from some protected session. This is indeed the motivation we have given for why forward secrecy is important. However, a session key need not be used to provide data confidentiality; it can be used for other security services such as data integrity and authentication, either alone or in combination with confidentiality. Although it may be less obvious, forward secrecy can also be applied to protect against data manipulation, fabricating the record of what happened in the past.

In the literature the distinction between the terms "forward secrecy" and "forward security" is often blurred. Indeed it is common to find papers which use these terms interchangeably as though their meaning is identical. Historically, forward secrecy was first used with respect to the key exchange literature, and forward security was used with respect to other, mainly public-key, primitives such as encryption and signatures. In this paper we take the view that *forward secrecy* is the same thing as *forward security for key exchange*. Thus when we use the term forward secrecy we refer only to key exchange protocols, but when we use the term forward security (without qualification) then we are being more general and may be thinking of any security goal.

In this section we consider asymmetric (public-key) primitives, which are more prominent in the literature when it comes to forward security. However, notions of forward security for symmetric-key cryptography have also been developed [6] and we will include them in our categorisation in Section 5. We may also generalize forward security to more properties than just confidentiality and authentication. An interesting case is the Tor protocol [22], which uses session keys to provide *unlinkability* of communication circuits. A forward security goal is that unlinkability should be preserved if nodes are compromised after the circuits are no longer in use.

Forward security for specific security goals can be achieved using dedicated primitives. We look at some important examples next, but first we note that when a session key is established using a key exchange protocol with forward secrecy, the protocol automatically enables forward security of any desired security property which relies on that session key. This is because when the session key is deleted at the end of the session, whatever security property it was providing can no longer be provided. For example, we can conceive of a property, which we might call *forward authenticity*, preventing messages from being fabricated in an earlier communications session, even under compromise of the participants. If the session key was used to provide message authenticity in the session, then forward authenticity is automatically provided if the session key is set up in a key exchange protocol with forward secrecy.

3.1 Forward-Secure Encryption

The notion of forward-secure encryption is attributed by Anderson [2] to Adam Back, who described it on an online mailing list in 1996. Forward-secure encryption has the goal to protect confidentiality of previously encrypted messages from being revealed when an entity is compromised later.

As proposed by Back, time is divided into intervals or *epochs*, so that a new private decryption key is used in each epoch, and the decryption key from the previous epoch is deleted. Of course this means that a different, or variant, public encryption key must also be used in each epoch. It is not difficult to design forward-secure encryption schemes in the case that the parameters of the scheme grow large quickly. Back proposed a scheme where the private key,

k_i, used in epoch i is derived from that in the previous epoch, $i-1$, via a one-way function H computing $k_i = H(k_{i-1})$. The corresponding public keys can then all be published and used in the correct epoch. This solution has fixed size private keys and ciphertexts, but the public key size increases linearly in the number of epochs which may not be acceptable.

Since this first proposal, cryptographers have proposed new and more practical schemes. Finding a satisfactory balance between the sizes of private keys, public keys and ciphertexts, is the main problem in designing good forward-secure encryption schemes. Formal models and new constructions were proposed by Canetti *et al.* [8, 9] who achieved schemes where each of private keys, public keys and ciphertexts increase in size only logarithmically in the number of epochs. Subsequently there are many more recent schemes with additional properties.

3.2 Forward-Secure Signatures

Anderson [2] seems to have been the first to suggest that authentication, in the form of digital signatures, can also benefit from forward security. He proposed *forward-secure signatures* which have the property that if the signer of a message is compromised by an adversary, then it should not be possible for the adversary to forge a signature from an earlier time. Authentication and confidentiality are in some senses dual concepts. Confidentiality is about protecting an adversary from reading, while authentication is about preventing an adversary from writing. Thus forward security for authentication should mean that an adversary who compromises a party now, should not be able to construct a message that was (apparently) authenticated at an earlier time.

As with forward-secure encryption, Anderson's forward-secure signatures achieve forward security by dividing time into epochs and updating, using a one-way transformation, the private key in each epoch. Due to the one-way nature of the updating algorithm, when the signer gets compromised, the signing key from the current epoch is leaked to the adversary, but signing keys from earlier epochs cannot be recovered. For this to work, the verification process must include as an input the epoch that a signature came from. Later, forward-secure signatures were formalised by Bellare and Miner [4] and have since been extended to include many variants such as group signatures [14, 52], ring signatures [49, 39], and blind signatures [13, 24].

3.3 Comparison

Forward-secure encryption and signatures share the same general goal with forward-secure key exchange of protecting past usage against current compromise. However, there is a significant difference in the ways that the examples in this section and the key exchange examples described in Section 2 achieve their goals. The latter use ephemeral keys and static long-term keys, while forward-secure encryption and signatures use epochs and evolving keys. Two significant consequences of this difference are:

- forward-secure key exchange requires interactivity in the traditional setting, while forward-secure encryption does not;
- with forward-secure key exchange past sessions are protected as soon as they are completed, while for forward-secure encryption, past ciphertexts are only protected if they are in an earlier epoch.

It is interesting to consider how the argument for why interaction is required, outlined at the end of Section 2, breaks down when considering forward-secure encryption. The difference is that in the traditional view the long-term secret is fixed, while in forward-secure encryption the long-term key evolves (or, equivalently, it holds state). In some deployments it may be undesirable to have to update the long-term key regularly; for example, the long-term key could be stored in a special hardware device, a hardware security module, with a limited interface and requiring special permission to write to. However, in general there seems no reason why the long-term key could not be updated. We now look at some more modern examples of how to update long-term keys in alternative ways.

4 Forward Secrecy in a Non-Interactive Setting

So far we have seen that forward secrecy is a property that is increasingly desirable but has been traditionally achieved with interactive protocols. Despite the huge advances in technology, higher efficiency and reduced delay are still highly sought after. Furthermore there are several prominent applications in which interaction between endpoints may not be practical. One example is electronic mail – we cannot assume that the recipient is online at all times. Even with interactive messaging, there is a bootstrapping problem when communicating for the first time. A third example is when sharing encrypted files in a cloud storage scenario; the decryption key must be shared, but recipients may not be online.

Thus there is need to achieve forward secrecy without interaction in modern communications. This need has been recognised and two conceptually different solutions have been proposed both in the academic literature and in real-world implementations. In this section we look at both approaches to this problem and then discuss how the properties that they achieve differ from using interactive protocols.

4.1 Puncturable Encryption

Puncturable encryption is a recently developed cryptographic primitive that was formally introduced by Green and Miers in 2015 [27]. Interestingly, Anderson has already given an informal description of puncturable encryption many years before [2]. Throughout the last years, several different constructions of protocols based on (improved) puncturable encryption have been proposed [29, 19, 18, 20, 53].

The core idea of puncturable encryption is to modify the private key of a scheme after each decryption. To be more precise, when a ciphertext c is decrypted by computing $\mathsf{Dec}(sk, c)$, we additionally replace the private key with the output of a puncture procedure $sk' \leftarrow \mathsf{Punct}(sk, c)$. This modified private key sk' will not be able to decrypt ciphertext c anymore, ensuring that c cannot be decrypted after the punctured key has been compromised. By repeatedly invoking the puncture procedure it is possible to stepwise revoke decryption capabilities from a private key.

Thus puncturable encryption provides a kind of forward security for public key encryption – compromise of the recipient's key does not compromise messages previously decrypted. Note, however, that we only enjoy this form of protection if c is *received and processed* by the recipient.

We can easily transform a puncturable encryption scheme to form a puncturable key exchange scheme as illustrated in Figure 3. The session key, k, is simply chosen by party A, encrypted using the puncturable encryption scheme, and sent to B. Now A does not have to wait to send a message protected by k, but can start sending user data immediately, even piggy-backed with the key exchange message. Note that this basic construction does not provide any authentication to B, so in many applications we will require additional mechanisms, for example a signature from A, in order to obtain a secure key exchange protocol.

In contrast to the fine-grained puncturing mechanism, sometimes a puncturable encryption scheme is also equipped with a coarse-grained mechanism to revoke decryption capabilities. This mechanism is inspired by the kind of forward-secure encryption explained in Section 3.1,

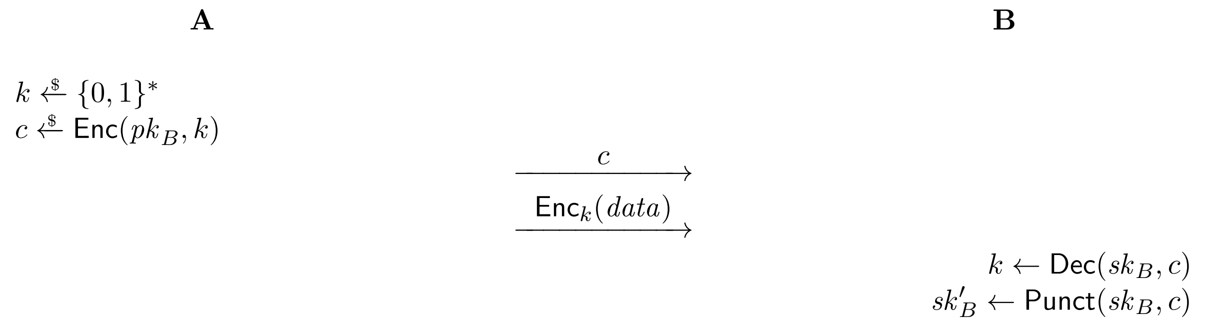
A
B
$k \xleftarrow{\$} \{0,1\}^*$
$c \xleftarrow{\$} \mathsf{Enc}(pk_B, k)$
$\xrightarrow{c}$
$\xrightarrow{\mathsf{Enc}_k(data)}$
$k \leftarrow \mathsf{Dec}(sk_B, c)$
$sk'_B \leftarrow \mathsf{Punct}(sk_B, c)$

Figure 3: Key exchange from puncturable encryption.

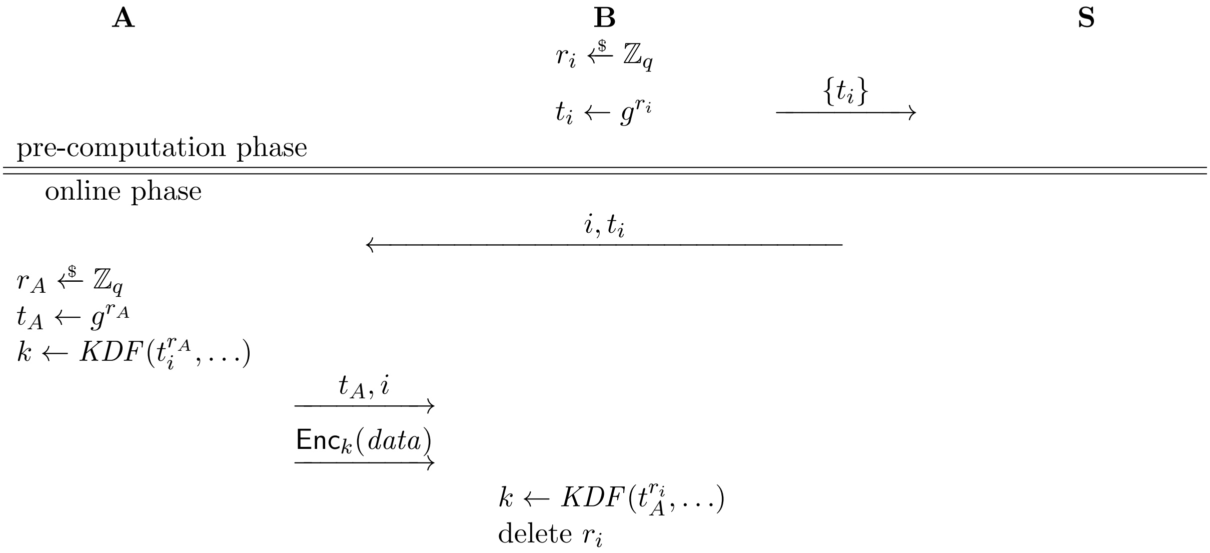
A
B
S
$r_i \xleftarrow{\$} \mathbb{Z}_q$
$t_i \leftarrow g^{r_i}$
$\xrightarrow{\{t_i\}}$
pre-computation phase
online phase
$\xleftarrow{i, t_i}$
$r_A \xleftarrow{\$} \mathbb{Z}_q$
$t_A \leftarrow g^{r_A}$
$k \leftarrow KDF(t_i^{r_A}, \ldots)$
$\xrightarrow{t_A, i}$
$\xrightarrow{\mathsf{Enc}_k(data)}$
$k \leftarrow KDF(t_A^{r_i}, \ldots)$
delete r_i

Figure 4: Key exchange with pre-computed keys.

and similarly divides time into epochs. After a certain time has passed, the private key will be "punctured" for an epoch that renders decryption of all messages sent in this epoch impossible. Note that this coarse approach ensures that even lost or intercepted messages enjoy protection.

Observe that for all puncturable encryption schemes the private key changes over time (while the public key remains static) and does not fit into the traditional model discussed in Section 2 where long-term keys remain unchanged during the system's lifetime.

4.2 Pre-Computed Keys

Suppose that party A wants to use Diffie–Hellman key exchange and then use the resulting session key to protect a message sent to party B. If B is not online to run the interactive Diffie–Hellman protocol then A could instead use some pre-computed value from B which has been sent to, and stored with, an online server S. This idea has been used in several real-world protocols. It is illustrated in Figure 4. In the pre-computation phase B generates some number of private/public keys pairs and stores the public keys with S. The online phase starts with party A retrieving an unused public key of party B from S.

The pre-computed key from party B, shown as t_i in Figure 4, is no longer an ephemeral key in the sense of existing only during one protocol run. As well as being pre-computed, it is also possible that t_i is *re-used* in different protocol runs and different instances of party B. This idea has been proposed at least as far back as 2004 for the JFK protocol [1]. In this case it may be desirable to impose a time limit on how long the t_i value should remain in use. This is what is done in the real-world protocol QUIC [12] where the value's lifetime may typically be two days.

In contrast, the real-world messaging protocol Signal [51], which uses pre-computed keys to protect the first communicated message in any conversation, intends that each pre-computed key is used only once. To allow this, the recipient B pre-computes many t_i values which are deleted after they have been used (including, more importantly, the corresponding private r_i values). However, since the number of new conversations that will be started is not predictable, Signal has a fallback mechanism to use the last available key for many conversations until the supply of pre-computed keys is replenished. Signal suggests keys be replenished once a week, or once a month [40].

Regardless of whether the pre-computed keys are used once or many times, the corresponding private keys r_i will have to be stored by B for some period before the protocol run actually takes place. This means that such keys are no longer ephemeral keys in the traditional view from Section 2. Are such keys then long-term keys? Some authors have given them the name *medium-term* keys. It is evident that such keys do not fit into the traditional model, but for sure they cannot become available to an adversary if we are to retain the forward secrecy property.

4.3 Message Suppression Attacks

Both aforementioned approaches to achieve forward secrecy without interaction are prone to a certain type of attack, which is not relevant in traditional key exchange models. We will call these attacks *message suppression attacks.* In a message suppression attack, the adversary intercepts and drops messages dedicated to the receiving party. If the adversary now compromises the receiving party at a later point in time, the adversary is able to decrypt the intercepted messages, seemingly breaking forward secrecy.

In puncturable encryption a compromise of the private key sk_B allows to decrypt intercepted messages of the form $(c, \mathsf{Enc}_k(data))$ if the message was never received by B and hence the private key sk_B is not punctured for c. A similar attack works when using pre-computed keys. Compromise of the private r_i values allows to decrypt intercepted messages of the form $(t_A, i, \mathsf{Enc}_k(data))$ if the message was never received by B and hence the value r_i not deleted.

This attack can be mitigated if we apply a time-based mechanism to the protocol that ensures the private values are modified or deleted after a certain time has passed. If this mechanism is invoked after a message is suppressed but before compromise of receiver B takes place, the attack cannot be mounted. In the case of puncturable encryption this might be a coarse-grained transformation of the receiver's private key that only allows decryption of messages sent in a certain time period. For pre-computed keys this could be a lifetime associated with the private values r_i.

Is a message suppression attack realistic? We first note that in formal security models for key exchange it is typically assumed that the adversary has complete control of the network, and message transmissions only occur when explicitly demanded by the adversary. Such an adversary can then easily engineer message suppression attacks. However, we also note that most security models cannot actually capture message suppression attacks. The adversary's formal security goal is to choose a protocol run (called a session) at a party which has accepted a session key k and then distinguish whether it is given k or a completely random string. But in a message suppression attack no (relevant) session actually exists at the receiver B, so the adversary is unable to choose it.

Green and Miers [27] explicitly mention the message suppression attack and use it as one motivation (the other being efficiency) to combine their fine-grained puncturable encryption scheme with coarse-grained forward-secure encryption. However, their security notion for puncturable encryption (IND-PUN-xxx) explicitly requires the tag(s) used for the challenge ciphertext to be already punctured when compromise takes place. This is essentially the same for their combined scheme security notion (IND-PFSE-xxx). Thus message suppression attacks are not covered by their notions.

4.4 Malicious Key Exhaustion

We remark that it is possible to maliciously exhaust key material in both aforementioned approaches by flooding the receiver with initiating messages (cf. Figures 3 and 4). If the key material is limited, this might lead to a situation where the recipient is unable to process incoming messages. This especially affects puncturable encryption schemes where correct decryption of messages can only be guaranteed for a polynomial number of punctures, such as the schemes by Derler *et al.* [19, 18], or all schemes storing only a polynomial number of pre-computed keys. Due to the non-interactive nature of the key exchange it is also difficult to avoid this attack; the recipient has no means of contributing fresh input.

In the case of pre-computed keys, Signal [51] acknowledges this attack vector and stores the last remaining pre-computed key if all others have been exhausted, sacrificing the single-use aspect of precomputed keys. The last precomputed key is then used for all following key exchanges until the recipient restocks its supply of pre-computed keys. Note that this approach does affect forward security. All sessions established with the last pre-computed keys can be compromised by an adversary until the recipient replenishes its stock of pre-computed keys.

5 Classifying Forward Security

Having seen various examples of forward security we now examine the fundamental similarities and differences between different schemes and identify how we may categorise them.

5.1 Dynamic Keys

We start by observing that in any kind of forward security it is essential that keying material is *dynamic*. A notion of time is inherent in defining forward security since we need to distinguish events that are past when compromise happens in the present. An adversary who compromises some party A is assumed to obtain the secrets of the compromised party. In order to prevent the adversary from performing exactly the same actions as A did in the past, something must have changed in the key material, even if it is only the deletion of generated randomness.

Note that it is possible to differentiate between different key types and define different forms of compromise. Indeed, many formal security models in the literature do this. Key exchange models [17] often allow adversaries (conditional) access to different oracles which reveal either long-term keys or randomness. There are good reasons why such differentiation may be realistic; for example, a long-term key may be stored in a secure physical device less accessible to the adversary. In relation to the traditional view from Section 2, *long-term* can be generalised to simply mean a key that can be compromised.

The strength of an adversary is determined by which type of keys it can compromise. For example, an adversary could have the ability to compromise secret values that are only stored in volatile memory for a short time (e.g., by mounting cold boot attacks [30]). A different adversary could only be able to access key which are stored in physical memory for several weeks. Of course, even adversaries able to perform both attacks could exist.

In any case, keys that never become available to the adversary do not influence the definition of forward security since all security guarantees provided by the protocol are still given. Note that an adversary that is able to predict the randomness used in the key derivation (e.g., backdoored random number generators, or bad randomness), may be able to break forward security by re-deriving the key without compromising any party; in general, forward security does not protect against bad randomness.

In the following we will first consider an adversary that is not able to compromise all key material, that is, we take a traditional path and distinguish between compromisable and non-compromisable key material. At the end of this section we will discuss the implications of adversaries that are able to compromise all key material.

Classes of Forward Security. From the examples we have seen, we can deduce different categories of forward security. The categories are parametrised via a parameter τ that defines the period in which the adversary can obtain, via compromise, the same keying material as originally used by the parties. This period of vulnerability starts from time 0 when the key is first defined and extends up to time τ. We identify three categories as follows:

Absolute Forward Security: $\tau = 0$. In this case the adversary has no opportunity to recover the keying material necessary to break security.

Delayed Forward Security: $0 < \tau < \infty$. In this case the adversary is able to break forward security if it is able to get access to the keying material before time τ (e.g., forward-secure encryption).

Null Forward Security: $\tau \to \infty$. In this case the adversary is able to break forward security by compromising at any time (e.g., the protocol shown in Figure 1).

Types of Keys. The three categories lead us to three different kinds of dynamic private keys. We identify and name them as follows:

volatile keys: which are never available to the adversary (e.g., Diffie–Hellman exponents);

windowed keys: which are available to the adversary for a finite period (e.g., keys with a lifetime);

triggered keys: which are available to the adversary until a defined protocol event occurs (e.g., pre-computed keys being used).

Note that traditional long-term keys are not included in these types since they are not dynamic. A protocol using only long-term keys will always have null forward security.

It is perhaps tempting to assume that these different key types are listed in an order of merit. The volatile keys, such as used in the Diffie–Hellman protocol, only exist while the protocol is being run, windowed keys may exist for some time after the protocol is run, while triggered keys exist before the protocol is run but should be deleted immediately afterwards. However, in practice the length of time that windowed and triggered keys remain vulnerable depends on the implementation details. In particular, windowed keys have a lifetime determined by the epoch length which may be chosen to be long or short in different applications while triggered keys may be stored longer than intended when a message suppression attack is launched.

We provide an overview of which class of forward security can be achieved, depending on the used key type and whether message suppression attacks are feasible, in Table 1. The last row in the table considers keys whose deletion can be triggered by an event, but will anyway

Key Type	*Message Suppression Attacks*	
	infeasible	*feasible*
volatile	absolute	absolute
windowed	delayed	delayed
triggered	absolute	null
windowed + triggered	absolute	delayed

Table 1: The achievable category of forward security depending on the key types and whether message suppression attacks are possible if volatile keys *cannot* be compromised.

be deleted at the end of some window. We have seen such keys in the description of the Green and Miers fine-grained and coarse-grained security described in Section 4.1. In practice we can expect that "pure" triggered keys are never used, but always have some lifetime window.

In existing security models for key exchange these key types are not differentiated in any way. Indeed forward secrecy definitions are easily adapted to fit all of these key types by simply stating that the adversary is prohibited from gaining access to them in any target session. This hides the significant practical differences between them, which have been frequently acknowledged in the literature [8, 9, 27]. These different key types can, in principle at least, be used to provide different classes of forward-secure versions of any cryptographic primitives.

5.2 Categorising Schemes

Table 2 gives examples of schemes which apply the different categories of keys. It turns out that for most primitives, each of the three categories of volatile, windowed and triggered keys can be instantiated with concrete schemes already existing in the literature. This gives us confidence that these are meaningful classes of keys. Moreover, we are not aware of schemes which claim to provide forward security but do not use keys in one of these classes, giving us optimism that the classification may be complete. Indeed, we suggest that any cryptographic property can naturally have a forward-secure version, by simply stating that its security properties should still hold when compromise happens later, where the meaning of "later" depends on the key type. To illustrate that they can be much more general we include session resumption and private circuit protocols as additional examples.

Asymmetric Encryption. For the case of asymmetric encryption, windowed and triggered keys have been examined earlier in the instances of forward-secure encryption (cf. Section 3.1) and puncturable encryption (cf. Section 4.1). It may seem that volatile keys are never needed for encryption, but one difference between forward-secure encryption and forward-secure key exchange is that in the former we do not require the sender (encryptor) to have a long-term or certified public key. Thus to achieve forward-secure encryption with volatile keys the sender and recipient can perform a Diffie–Hellman exchange with only the receiving party authenticating the ephemeral value, and then the Diffie–Hellman shared secret is used to encrypt. This construction has been proposed by Dodis and Fiore [23] and named *interactive encryption.*

Symmetric Encryption. As in the asymmetric encryption case discussed above, forward-secure encryption with volatile keys can be achieved with an interactive encryption protocol using the shared secret to authenticate a Diffie–Hellman key exchange, then deriving a session key to encrypt the message. Here it is always possible for both parties to authenticate their

Primitive	*Key Type*		
	volatile	*windowed*	*triggered*
Asymmetric encryption	Interactive encryption [23]	Epoch-based encryption [8, 9]	Puncturable encryption [27]
Symmetric encryption	Interactive encryption [23]	Bellare–Yee encryption [6]	Symmetric puncturable encryption [54]
Authenticated key exchange	Signed Diffie–Hellman [44]	Epoch-based Diffie–Hellman [44]	Pre-keyed Diffie–Hellman [51]
Digital signatures		Forward-secure signatures [2, 4]	Puncturable signatures [59]
Session resumption protocols	TLS 1.3 PSK-(EC)DHE mode [47]	STEK rotation [38]	0-RTT session resumption protocols [3]
Circuit construction protocols	nTor [26]	NI-OR [10]	T0RTT [37]

Table 2: Example schemes in different categories.

Diffie–Hellman ephemeral key using the shared key, but for encryption it is necessary only that the recipient authenticates.

Bellare and Yee [6] construct a key-evolving symmetric encryption scheme by updating the key in each epoch using what they call a forward-secure pseudorandom bit generator. This achieves windowed forward secrecy. We note, in passing, that the same authors also define and instantiate windowed forward-secure message authentication. A notion of symmetric key puncturable encryption has been defined by Sun *et al.* [54] which provides triggered forward-secure symmetric encryption.

Authenticated Key Exchange. When we think of forward security for key exchange, we typically first think of interactive Diffie–Hellman which uses volatile keys. Key exchange from puncturable encryption, explained in Section 4.1, and key exchange with pre-computed keys, explained in Section 4.2, both achieve forward-secure key exchange with triggered keys. Pointcheval and Sanders [44] give a construction for windowed key exchange for non-interactive key exchange, but this same construction could be adapted to provide windowed authenticated key exchange.

Digital Signatures. Forward security for signatures is easier to achieve than for encryption and key exchange in the sense that the signer can update the dynamic keying material without needing to wait to receive anything from the verifier. Because of this, signatures with windowed keys can be simply adapted so that they update the relevant dynamic keys immediately upon signing any message, so that the keys become triggered. In addition, and for the same reason, suppression attacks are not relevant for forward-secure signatures so that the right-hand column in Table 1 does not apply to signatures. Thus using triggered keys for signatures always gives absolute forward security. Puncturable signatures, the analogue of puncturable encryption using triggered keys, have also found uses [59]. There does not seem any obvious way to achieve signatures with volatile keys and, to our knowledge, no such primitive has been proposed in the literature.

Session Resumption Protocols. Another interesting case related to key exchange is that of session resumption protocols. In such protocols a client wants to resume a previous session with a server using a shared (authenticated) secret, which has been established in a previous session. A widely used session resumption protocol is, for example, the pre-shared key (PSK) mode in TLS 1.3 [47]. Since session resumption is always tied to a previously established secret, we need to carefully evaluate whether forward security is actually achieved.

Similar to key exchange we are able to achieve forward security with volatile keys when executing an additional Diffie–Hellman key exchange, and switching to the Diffie–Hellman key, after resumption has taken place. This idea is deployed in the PSK-EC(DHE) mode in TLS 1.3. Note, that this approach requires interactivity and does not ensure forward security for any messages solely protected under the pre-shared secret if the pre-shared secret gets compromised.

Alternatively, a concept known as *STEK rotation* can be deployed. In this case the server maintains a dedicated symmetric key, sometimes called session ticket encryption key (STEK), which is used to encrypt the shared secret. The ciphertext $c = \mathsf{Enc}(stek, secret)$ is stored at the client while the server is able to delete both c and *secret*. In order to resume the session, the client simply sends c back to the server. Forward security with windowed keys is achieved if the STEK is replaced regularly; for example, Cloudflare deploys this approach and rotates their keys roughly once a day [38].

Recent work by Aviram *et al.* [3] shows that it is also possible to achieve forward security for session resumption with triggered keys. They utilise so-called *puncturable pseudorandom functions*. The main idea is to compute session keys by evaluating a pseudorandom function. Once the session key is computed, the pseudorandom function's instantiation will be altered in such a way that recomputation of the session key is impossible.

Circuit Construction Protocols. Circuit construction protocols are anonymity-providing multi-party protocols executed between an initiator and several servers. The initiator picks a subset (typically three) of available servers and establishes a session key with each server such that it is oblivious to an observing adversary which servers have been picked by the initiator. The established keys can then be used in a so-called *routing protocol* to communicate in a secure and anonymous way. Defining security for circuit construction protocols is not trivial, and hence we will only give a brief intuition of forward security and refer the reader to literature that addresses circuit construction protocols otherwise [10, 37, 26].

The notion of forward security for circuit construction protocols is much more complex compared to the notion of key exchange. Besides providing key indistinguishability (as known from standard key exchange), forward security also captures a notion of anonymity preservation. To be more precise, the compromise of multiple servers should not endanger the initiator's anonymity in any way. This property is sometimes termed cryptographic unlinkability – as long as circuit construction includes one honest server, an adversary should not be able to link connections, even if all servers get compromised after the session is closed.

Currently, the nTor [26] protocol is the most widely adopted protocol for circuit construction. It performs several executions of the Diffie–Hellman protocol between multiple parties and achieves forward security with volatile keys. In order to reduce latency, Catalano *et al.* [10] have proposed a non-interactive circuit construction protocol based on forward-secure encryption that achieves forward security with windowed keys. Recently, Lauer *et al.* [37] proposed a non-interactive circuit construction protocol utilising puncturable encryption, achieving forward security with triggered keys.

Key Type	*Message Suppression Attacks*	
	infeasible	*feasible*
volatile	delayed*	delayed*
windowed	delayed**	delayed**
triggered	delayed*	null
windowed + triggered	delayed*	delayed**

Table 3: The achievable category of forward security depending on the key types and whether message suppression attacks are possible if *any* secret value can be compromised. * with $\tau =$ session length; ** with $\tau = \max\{\text{window}, \text{session length}\}$.

5.3 Stronger Adversaries

In the previous paragraphs we have assumed that it is possible to implement protocols using volatile keys which are never available to the adversary. This may be realistic, for example when the only way for an adversary to compromise a party is by enforcing a legal order and even a cooperating party is unable to extract keys from volatile memory. In this paragraph we discuss the implications of an adversary who can obtain any key that exists. To be more precise, we now assume that an adversary is even able to compromise volatile keys as well as retrieve keys that only exist in volatile memory. This induces the following changes in comparison with Table 1.

Any protocol that uses volatile keys provides forward security only after the session has expired, hence only achieving delayed forward security. Similarly, triggered keys cannot achieve absolute forward security as they need to be kept in volatile storage until the session terminates, making them vulnerable to compromise as well. Instead triggered keys are only able to provide delayed forward security after the session has expired. Table 3 shows the achievable level of forward security for each key type.

6 Conclusion

Our goal in this paper has been to put forward security and forward secrecy into a modern context. We have tried to show that they are related by proposing that forward secrecy is the same as forward security for key exchange, even though not all authors agree on this. Furthermore, we have shown how the concept of forward secrecy can be generalised to many other cryptographic primitives. We also identified different categories of forward security and have shown which of them are achieved in several schemes using a variety of techniques.

We have pointed out that our different forward security concepts have not been captured in existing formal security models. It may be interesting to try to do this and show formally that certain primitives have the same, or different, levels of forward security.

Post-compromise security [16] is in some sense the dual concept of forward security, since it is concerned with obtaining security for the future in the face of compromise of parties now. It could be interesting to consider whether our categories of forward security have analogues for post-compromise security which could perhaps be incorporated into a unified model.

References

[1] William Aiello, Steven M. Bellovin, Matt Blaze, Ran Canetti, John Ioannidis, Angelos D. Keromytis, and Omer Reingold. Just fast keying: Key agreement in a hostile internet.

ACM Trans. Inf. Syst. Secur., 7(2):242–273, 2004.

[2] Ross Anderson. Two remarks on public key cryptology. Technical report, University of Cambridge, Computer Laboratory, 2002. `https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-549.pdf`.

[3] Nimrod Aviram, Kai Gellert, and Tibor Jager. Session resumption protocols and efficient forward security for TLS 1.3 0-RTT. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 117–150, Darmstadt, Germany, May 19–23 2019. Springer, Heidelberg, Germany.

[4] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 431–448, Santa Barbara, CA, USA, August 15–19 1999. Springer, Heidelberg, Germany.

[5] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany.

[6] Mihir Bellare and Bennet S. Yee. Forward-security in private-key cryptography. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *LNCS*, pages 1–18, San Francisco, CA, USA, April 13–17 2003. Springer, Heidelberg, Germany.

[7] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 321–336, Amsterdam, The Netherlands, April 28 – May 2 2002. Springer, Heidelberg, Germany.

[8] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 255–271, Warsaw, Poland, May 4–8 2003. Springer, Heidelberg, Germany.

[9] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. *Journal of Cryptology*, 20(3):265–294, July 2007.

[10] Dario Catalano, Mario Di Raimondo, Dario Fiore, Rosario Gennaro, and Orazio Puglisi. Fully non-interactive onion routing with forward secrecy. *Int. J. Inf. Secur.*, 12(1):33–47, February 2013.

[11] Dario Catalano, Dario Fiore, and Rosario Gennaro. Certificateless onion routing. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM CCS 2009*, pages 151–160, Chicago, Illinois, USA, November 9–13 2009. ACM Press.

[12] Wan-Teh Chang and Adam Langley. QUIC crypto, 2014. `https://docs.google.com/document/d/1g5nIXAIkN\_Y-7XJW5K45IblHd\_L2f5LTaDUDwvZ5L6g`.

[13] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.

[14] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265, Brighton, UK, April 8–11 1991. Springer, Heidelberg, Germany.

[15] Liqun Chen and Caroline Kudla. Identity based authenticated key agreement protocols from pairings. In *16th IEEE Computer Security Foundations Workshop (CSFW-16)*, pages 219–233. IEEE Computer Society, 2003.

[16] Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt. On post-compromise security. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 164–178. IEEE Computer Society, 2016.

[17] Cas Cremers. Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong, editors, *ASIACCS 11*, pages 80–91, Hong Kong, China, March 22–24 2011. ACM Press.

[18] David Derler, Kai Gellert, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Bloom filter encryption and applications to efficient forward-secret 0-RTT key exchange. Cryptology ePrint Archive, Report 2018/199, 2018. `https://eprint.iacr.org/2018/199`.

[19] David Derler, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Bloom filter encryption and applications to efficient forward-secret 0-RTT key exchange. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 425–455, Tel Aviv, Israel, April 29 – May 3 2018. Springer, Heidelberg, Germany.

[20] David Derler, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. I want to forget: Fine-grained encryption with full forward secrecy in the distributed setting. Cryptology ePrint Archive, Report 2019/912, 2019. `https://eprint.iacr.org/2019/912`.

[21] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[22] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[23] Yevgeniy Dodis and Dario Fiore. Interactive encryption and message authentication. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 494–513, Amalfi, Italy, September 3–5 2014. Springer, Heidelberg, Germany.

[24] Dang Nguyen Duc, Jung Hee Cheon, and Kwangjo Kim. A forward-secure blind signature scheme based on the strong RSA assumption. In Sihan Qing, Dieter Gollmann, and Jianying Zhou, editors, *ICICS 03*, volume 2836 of *LNCS*, pages 11–21, Huhehaote, China, October 10–13 2003. Springer, Heidelberg, Germany.

[25] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 254–271, Nara, Japan, February 26 – March 1 2013. Springer, Heidelberg, Germany.

[26] Ian Goldberg, Douglas Stebila, and Berkant Ustaoglu. Anonymity and one-way authentication in key exchange protocols. *Designs, Codes and Cryptography*, 67, 02 2012.

[27] Matthew D. Green and Ian Miers. Forward secure asynchronous messaging from puncturable encryption. In *2015 IEEE Symposium on Security and Privacy*, pages 305–320, San Jose, CA, USA, May 17–21 2015. IEEE Computer Society Press.

[28] Christoph G. Günther. An identity-based key-exchange protocol. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *EUROCRYPT'89*, volume 434 of *LNCS*, pages 29–37, Houthalen, Belgium, April 10–13 1990. Springer, Heidelberg, Germany.

[29] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-RTT key exchange with full forward secrecy. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 519–548, Paris, France, April 30 – May 4 2017. Springer, Heidelberg, Germany.

[30] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In Paul C. van Oorschot, editor, *USENIX Security 2008*, pages 45–60, San Jose, CA, USA, July 28 – August 1 2008. USENIX Association.

[31] ISO. IT security techniques – key management – part 2: Mechanisms using symmetric techniques, January 2018. `https://www.iso.org/standard/73207.html`.

[32] Subodh Iyengar and Kyle Nekritz. Building zero protocol for fast, secure mobile connections, 2017. `https://code.fb.com/android/building-zero-protocol-for-fast-secure-mobile-connections/`.

[33] Ik Rae Jeong, Jonathan Katz, and Dong Hoon Lee. One-round protocols for two-party authenticated key exchange. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *ACNS 04*, volume 3089 of *LNCS*, pages 220–232, Yellow Mountain, China, June 8–11 2004. Springer, Heidelberg, Germany.

[34] Aniket Kate, Greg M. Zaverucha, and Ian Goldberg. Pairing-based onion routing with improved forward secrecy. *ACM Trans. Inf. Syst. Secur.*, 13(4):29:1–29:32, December 2010.

[35] Werner Koch. The GNU privacy guard. `https://gnupg.org/`.

[36] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566, Santa Barbara, CA, USA, August 14–18 2005. Springer, Heidelberg, Germany.

[37] Sebastian Lauer, Kai Gellert, Robert Merget, Tobias Handirk, and Jrg Schwenk. T0RTT: Non-interactive immediate forward-secret single-pass circuit construction. *Proceedings on Privacy Enhancing Technologies*, 2020(2):336–357, 2020.

[38] Zi Lin. TLS Session Resumption: Full-speed and Secure, 2015. `https://blog.cloudflare.com/tls-session-resumption-full-speed-and-secure/`.

[39] Joseph K Liu and Duncan S Wong. Solutions to key exposure problem in ring signature. *IJ Network Security*, 6(2):170–180, 2008.

[40] Moxie Marlinspike and Trevor Perrin. The X3DH key agreement protocol, 2016. `https://signal.org/docs/specifications/x3dh/`.

[41] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the Association for Computing Machinery*, 21(21):993–999, December 1978.

[42] Lasse Øverlier and Paul F. Syverson. Improving efficiency and simplicity of tor circuit establishment and hidden services. In Nikita Borisov and Philippe Golle, editors, *PET 2007*, volume 4776 of *LNCS*, pages 134–152, Ottawa, Canada, June 20–22 2007. Springer, Heidelberg, Germany.

[43] DongGook Park, Colin Boyd, and Sang-Jae Moon. Forward secrecy and its application to future mobile communications security. In Hideki Imai and Yuliang Zheng, editors, *PKC 2000*, volume 1751 of *LNCS*, pages 433–445, Melbourne, Victoria, Australia, January 18–20 2000. Springer, Heidelberg, Germany.

[44] David Pointcheval and Olivier Sanders. Forward secure non-interactive key exchange. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 21–39, Amalfi, Italy, September 3–5 2014. Springer, Heidelberg, Germany.

[45] Blake Ramsdell. Secure/multipurpose internet mail extensions (S/MIME) version 3.1 certificate handling, July 2004. RFC3850.

[46] Blake Ramsdell. Secure/multipurpose internet mail extensions (S/MIME) version 3.1 message specification, July 2004. RFC3851.

[47] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, IETF, August 2018.

[48] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

[49] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565, Gold Coast, Australia, December 9–13 2001. Springer, Heidelberg, Germany.

[50] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, October 1997.

[51] Signal Messenger. Signal messenger website, 2019. `https://signal.org/`.

[52] Dawn Xiaodong Song. Practical forward secure group signature schemes. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 225–234, Philadelphia, PA, USA, November 5–8 2001. ACM Press.

[53] Shi-Feng Sun, Amin Sakzad, Ron Steinfeld, Joseph Liu, and Dawu Gu. Public-key puncturable encryption: Modular and compact constructions. PKC 2020, 2020. To appear.

[54] Shifeng Sun, Xingliang Yuan, Joseph K. Liu, Ron Steinfeld, Amin Sakzad, Viet Vo, and Surya Nepal. Practical backward-secure searchable encryption from symmetric puncturable encryption. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 763–780, Toronto, ON, Canada, October 15–19 2018. ACM Press.

[55] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. SoK: Secure messaging. In *2015 IEEE Symposium on Security and Privacy*, pages 232–249, San Jose, CA, USA, May 17–21 2015. IEEE Computer Society Press.

[56] Jianghong Wei, Xiaofeng Chen, Jianfeng Wang, Xuexian Hu, and Jianfeng Ma. Forward-secure puncturable identity-based encryption for securing cloud emails. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *Computer Security – ESORICS 2019*, volume 11736 of *LNCS*, pages 134–150, Luxembourg, September 23–27 2019. Springer International Publishing.

[57] Nico Williams. [TLS] 0-RTT security considerations (was OPTLS), 2014. `https://mailarchive.ietf.org/arch/msg/tls/OZwGgVhySbVhU36BMX1elQ9xOGE`.

[58] David J. Wu, Ankur Taly, Asim Shankar, and Dan Boneh. Privacy, discovery, and authentication for the internet of things. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016, Part II*, volume 9879 of *LNCS*, pages 301–319, Heraklion, Greece, September 26–30 2016. Springer, Heidelberg, Germany.

[59] Mark Zhandry. How to avoid obfuscation using witness PRFs. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 421–448, Tel Aviv, Israel, January 10–13 2016. Springer, Heidelberg, Germany.

A Glossary of Terms

In this section we provide a description of terms that have been used in literature to describe aspects of forward security.

Asynchronous Forward Security/Secrecy: Rather than describing a particular property, these terms have been used to refer to forward security/secrecy in asynchronous communications [55]. This is the same concept as what we have called non-interactive protocols throughout this paper.

Eventual Forward Secrecy: This term is used in the context of circuit construction protocols and was coined by Øverlier and Syverson [42]. They define that eventual forward secrecy is achieved if forward secrecy is achieved a certain time period after the circuit has been closed. We remark that the term “secrecy” might be confusing in this context as circuit construction protocols typically demand more security guarantees such as maintaining the privacy of a circuit. Hence, we believe that “eventual forward security” captures the same notion if forward security is seen as a generalisation of forward secrecy (cf. Section 3). Furthermore, the definition aligns with our category of “delayed forward security” (cf. Section 5).

Full Forward Secrecy: This term was used by Günther *et al.* [29] informally as a means to emphasise their claim that their proposed scheme, using puncturable encryption, does not suffer any deficiency in comparison with other forms of forward secrecy. Thus it is simply a synonym for forward secrecy.

Immediate Forward Secrecy: This term is used in the context of circuit construction protocols and was coined by Øverlier and Syverson [42]. They define that immediate forward secrecy is achieved if forward secrecy is achieved immediately after the circuit has been closed. We remark that the term “secrecy” might be confusing in this context as circuit construction protocols typically demand more security guarantees such as maintaining the privacy of a circuit. Hence, we believe that “immediate forward security” captures the same notion if forward security is seen as a generalisation of forward secrecy (cf. Section 3).

Furthermore, the definition aligns with our category of "absolute forward security" (cf. Section 5).

Non-Interactive Forward Security/Secrecy: This term was first used by Adam Back in an online posting (see discussion in Section 3.1). Like the term "asynchronous forward secrecy", this is a general concept, rather than a formal definition, in the same way that we have referred to non-interactive protocols throughout this paper. Although Back used the term "non-interactive forward secrecy", several later authors use "non-interactive forward security" as a synonym.

Partial Forward Security/Secrecy: When defining adversary capabilities we have implicitly assumed that any relevant party can be compromised after the target protocol session has completed. Some protocols can remain secure if the adversary is restricted to only compromising a subset of protocol parties, and such protocols are said to provide partial forward secrecy [43]. As a simple example, suppose we change the basic Diffie–Hellman protocol (Figure 2) so that one party, say A, uses a long-term public/private key pair instead of an ephemeral key pair. Then compromise of A will reveal the shared secret, while compromise of B, who continues to use an ephemeral key, will not. In this paper we have not considered partial forward secrecy, but the concept can be applied to many of the examples and definitions we have considered. Several authors used the term "partial forward security" as a synonym.

Perfect Forward Secrecy: This was the original term used by Günther in the first definition of forward secrecy [28] and many authors continue to use this term[3]. However, in common with other authors we prefer to drop the qualifier "perfect" on the grounds that it is redundant and potentially misleading due to the connotation with "perfect secrecy". The latter implies unconditional security which is usually not achieved with protocols providing forward secrecy.

Weak Forward Secrecy: This notion, first discussed by Bellare *et al.* [5], is achieved by a protocol if the adversary is disallowed from taking an active role in the target session. As pointed out by Krawczyk [36], this property is often achieved by two-message key exchange protocols even if a stronger notion is missing. This has been misinterpreted by many researchers to mean that no two-message protocol can achieve forward secrecy, but this is false [33].

Strong Forward Secrecy: The first usage of this term [7] seems to have been a form of forward secrecy where the adversary, upon compromise of a party, obtains not only long-term keys but also internal memory of the party. This meaning has been prominent in analysis of group key exchange and is similar to our stronger adversaries discussed at the end of Section 5. This term has also been used by some authors as a synonym for forward secrecy, to differentiate it from weak forward secrecy.

[3] Although Günther was writing in the context of identity-based encryption, the same terminology has continued to be used in the literature in the ordinary public key setting.