STRONG FORWARD SECURITY

Mike Burmester*
Department of Computer Science, Florida State University
214 Love Building, Tallahassee, Florida 32306, USA
burmester@cs.fsu.edu

Vassilios Chrissikopoulos
Department of Archiving and Library Studies, Ionian University
Corfu, 49100, GREECE
vchris@ionio.gr

Panayiotis Kotzanikolaou
Department of Informatics, University of Piraeus
80 Karaoli & Dimitriou, 185 34, Piraeus, GREECE
pkotzani@unipi.gr

Emmanouil Magkos*
Department of Informatics, University of Piraeus
80 Karaoli & Dimitriou, 185 34, Piraeus, GREECE
emagos@unipi.gr

Abstract Forward security has been proposed as a method to minimize the consequences of key exposure. In this paper we analyze this method and consider a vulnerability, which is due to the fact that the exposure may not have been detected. All forward secure cryptosystems proposed so far are vulnerable during the period between key exposure and its detection. We consider the notion of strong forward security in which cryptographically processed data is protected not only for the periods prior to key exposure but also after key exposure, and present two applications with this novel property: a basic public key cryptosystem and an ElGamal-based key escrow scheme.

Keywords: Forward security, key update, intrusion detection

*Research supported by the Secretariat of Research and Technology of Greece.

1. INTRODUCTION

A major security concern in every cryptosystem is the protection of secret keys from exposure. If the adversary appropriates the secret keys of a user in an encryption scheme, then the adversary can decrypt all ciphertexts intended for that user and confidentiality is lost. For a signature scheme, the adversary can masquerade as the legitimate user.

The problem of key exposure is critical in open environments such as the Internet, where every computer node is a potential victim of hackers. Thus, there is a need to adopt mechanisms that minimize the consequences of key exposure. So far, these mechanisms generally rely on secret distributed computation [9, 14, 15, 17, 22, 29], periodical key updating and key revocation [2, 5, 11, 20, 23, 25, 27].

Gunther [20] was the first to propose an encryption key updating mechanism that protects the confidentiality of all encrypted messages prior to key exposure. With this mechanism all encrypted material is protected from key exposure after the keys are updated. This property was called *forward secrecy.* With forward secrecy, disclosure of long-term secret keying material does not compromise the secrecy of earlier encrypted material [11, 20].

A solution that establishes forward secrecy in the context of real-time multicasting over large dynamic groups was proposed by McGrew and Sherman in [27]. Burmester, Desmedt and Seberry [5] proposed an escrow system with forward secrecy. There are also solutions that address the key exposure problem for digital signatures. Herzberg *et al* [22] consider threshold signature schemes (see also [9]) in which the users update their shares proactively. These schemes offer forward security, however the distribution of shares and the distributed computation required to compute signatures make them rather inefficient (*cf.* the discussion in [2]). Bellare and Miner [2] proposed efficient digital signatures with forward security, but their security can only be proven in the Random Oracle Model [3]. Recently, Krawczyk [25] proposed a solution that can be used with any signature scheme. In this paper we shall adopt the term *forward security* both for encryption and signatures.

There is an inherent weakness in forward security that follows from the fact that the definition does not specify what happens *after an intrusion*, when the secret information has been exposed to the adversary, and *until its detection*, when the public key is revoked. During this period the security of the system is compromised. For example, suppose that the adversary (*e.g.* a hacker) has appropriated the secret keys of Alice during the session t_e but the intrusion has not been detected (Fig. 1). The adversary will be able to update the stolen keys in the same way as

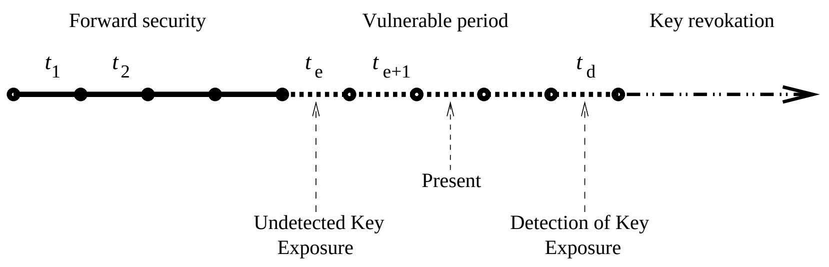

Figure 1 Forward Security

Alice and then generate secret keys for the sessions $t_{e+1}, \ldots, t_d$, until the intrusion is detected. This means that cryptographically processed data after key exposure is not protected. All forward secure schemes in the literature [2, 5, 20, 25] are vulnerable during this period. They only offer protection for sessions prior to key exposure.

Organization. In this paper we analyze forward security and consider a new threat in which the adversary appropriates *all* the secret keying material of a user without being detected. In Section 2 we consider the notion of strong forward security, in which cryptographically processed data is protected not only during the periods prior to key exposure but also during the periods after key exposure. In Section 3 we show how strong forward security can be achieved with any public key cryptosystem and in Section 4 we propose a strong forward secure key escrow/recovery scheme which is based on the ElGamal cryptosystem. We conclude in Section 5.

2. FROM FORWARD SECURITY TO STRONG FORWARD SECURITY

Suppose that Alice uses a forward secure cryptosystem and that the adversary has appropriated (all) her secret keying material during session t_e – see Figure 1. The adversary will not be able to obtain the keys for earlier sessions $t_j < t_e$, but will be able to update the key of session t_e in the same way as Alice, to get keys for sessions $t_{e+1}, \ldots,$ until session t_d, when the intrusion is detected. With the encryption scheme in [5], the updating is *deterministic* so the adversary will generate an identical key to Alice's, and thus decrypt all ciphertexts intended for Alice. A similar argument applies to the signature schemes in [2, 25]. In this case the adversary can forge Alice's signatures. With the encryption scheme in [20], which uses *randomized* updating, the adversary will generate a different key. However the adversary can prove that this key

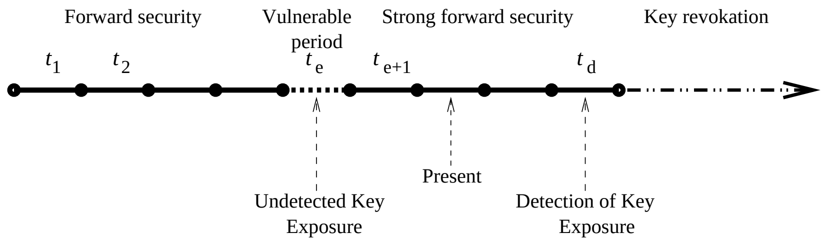

Figure 2 Strong Forward Security

is "genuine", since the adversary has also appropriated the long term authentication keys of Alice.

Regardless of whether the updating mechanism is deterministic or randomized, all cryptographically processed data is at risk during the period between key exposure and its detection. Protection from intrusions in which all the secret keying material of Alice is stolen can only be achieved by using non-cryptographic means. However, with randomized key updating this task should be easier, because Alice's updated key will be different from the key generated by the intruder (with high probability).

Definition. A system is *strongly* forward secure if disclosure of secret keying material does not compromise the security of the system for sessions both prior to exposure ($t_j < t_e$) and after exposure ($t_j > t_e$) – see Figure 2.

A practical but expensive solution. Strong forward security can be achieved with any public key cryptosystem by using threshold cryptography [9, 16, 17]. For this purpose the secret key is shared among several entities, which jointly execute the cryptographic application. The shares are then proactively updated [15, 23, 22]. Strong forward security is clearly achieved, provided that the threshold is sufficiently large.

With such schemes each application (encryption or digital signature) requires a distributed computation and therefore may be quite costly (as noted in [2]). Furthermore, the distribution of shares may be costly.

Our solution. Our goal is to achieve strong forward security in a practical and affordable way. The user must be able to certify new session keys with minimum cost, without out-of-band authentication. Furthermore, this should not involve costly distributed computations for each application (encryption or signature). For this purpose we combine randomized key updating with certification.

If a hacker appropriates the secret keying material of a legitimate user and then tries to certify an updated stolen key, then two valid public keys corresponding to the same user will be submitted for certification: the legitimate key and an alias key. The intrusion will be detected and thus the cryptographic security will only be compromised during the session of the intrusion.

3. A BASIC SOLUTION FOR ANY PUBLIC KEY CRYPTOSYSTEM

Based on our discussion above we can make any public key cryptosystem strongly forward secure. First let us consider digital signatures.

Suppose that the public/secret key pair of Alice for session t, is $(PK_{A,t}, SK_{A,t})$ and that $Cert(ID_A, PK_{A,t})$ is a certificate for it, issued by the Certifying Authority CA, where ID_A is a unique identifier of Alice. For the next session, Alice selects a random public/secret key pair $(PK_{A,t+1}, SK_{A,t+1})$, and digitally signs it together with ID_A, using her previous key: $sig_{SK_{A,t}}(ID_A, PK_{A,t+1})$. Alice then sends this together with her old certificate $Cert(ID_A, PK_{A,t})$ to the CA, which verifies Alice's signature using the old key $PK_{A,t}$. If this is correct, the CA sends Alice a new certificate $Cert(ID_A, PK_{A,t+1})$.

If an intruder appropriates (all) the secret keys of Alice during the session t (and in particular $SK_{A,t}$) and if the intruder submits an updated public key to the CA for certification, then two public keys will be submitted, both on behalf of Alice. If this happens the CA will revoke (all) the public keys of Alice.

A similar approach can be used for public key encryption. In this case however Alice needs two pairs of public keys, one for encryption and the other to authenticate her encryption key.

This basic scheme achieves strong forward security and is as secure as the underlying cryptosystem. Furthermore, it is very efficient. In particular, the certification of the public keys in each session does not require out-of-band methods. In addition, the size of keys and of the signatures does not expand as the keys are updated. However, we have a linear expansion in the number of certificates.

Remark 1. Although the protection of strong forward security is obvious in the case of encryption, one could argue that in the context of digital signatures it does not offer any additional protection to forward security. Consider for example the case when Bob has appropriated Alice's signing key. Then, even though Bob will not be able to update the stolen key without being detected, he could indirectly bypass the security of the system for future sessions. For example, he could sign postdated checks on behalf of Alice.

However, there are cases when strong forward security makes sense in the context of signatures. For example, when the lifetime of the signing key also restricts the scope of the signed message. This would make postdated checks (for later sessions) invalid.

Remark 2: "Imprisonment" attack. The proposed solution assumes that the attacker and the legitimate user have access to a Certifying Authority CA to update keys. This forces the attacker to "publish" the fact that a key has been exposed. If the attacker can somehow prevent the legitimate user from accessing the CA, then the attacker can impersonate the user for as long as he can confine the user. There seems to be no cryptographic way to handle such attacks.

4. AN ELGAMAL KEY ESCROW SCHEME WITH STRONG FORWARD SECURITY

The solution proposed above is not satisfactory for key escrow because the updated keys must be distributed among escrow agents (an excellent survey of key escrow systems is given by Denning and Branstad in [8]). The following scheme reduces the cost of key distribution and key updating by having the escrow agents regulate the timing process for key updating.

For simplicity, we describe a basic 2-out-of-2 key escrow scheme with escrow agents EA_1, EA_2, in which the Law Enforcement Agency LEA also acts as a Certifying Authority. The escrow agents and the LEA are trusted to adhere to the protocol.

Each user, say Alice, during setup, chooses a long-term secret key and shares this among the escrow agents in a verifiable way. Then, at the beginning of each session t the escrow agents select a time-control identifier h_t. This is broadcast by the LEA and will be used by all the users of the system for key updating. In particular, Alice will update her private key SK_{t-1} to SK_t by using her long-term secret key, some randomness and the time-control identifier h_t. After each updating, Alice and the escrow agents delete all information that might be useful to an adversary who may attempt to recover previous keys. Additionally, Alice updates her public key to PK_t, and proves to the LEA in zero-knowledge [19] that this has been properly constructed. The LEA then certifies the updated public key PK_t.

A hacker who succeeds in appropriating Alice's secret keying material may attempt to update the stolen session key and to get the updated key certified by the LEA. However, Alice will also submit her updated key for certification. The two keys are different (with overwhelming probability). The LEA will notice that different keys corresponding to the same user are submitted for certification, and thus detect the intrusion and revoke all the public keys of Alice.

Background. We use an ElGamal encryption scheme [12]. Let r, q, p be large primes with $q = 2r + 1$, $p = 2q + 1$, and let H be a subgroup of Z_q^* of order r with generator h, and G be a subgroup of Z_p^* of order q with generator g. For simplicity, and when there is no ambiguity, we drop the modulus operators. Also, we write $a \in_R A$ to indicate that the element a is chosen randomly with uniform distribution from the set A.

The Diffie-Hellman [10] operator DH is defined by $DH(g^a, g^b) = g^{ab}$. Given the numbers g^a and g^b, the problem of computing $DH(g^a, g^b)$ is called the *Diffie-Hellman* problem. The problem of deciding whether $z = DH(g^a, g^b)$, for a given $z \in Z_p$, is called the *Decision* Diffie-Hellman DDH problem [10].

Setup. Alice chooses a long term private key $x_A \in_R Z_q^*$ and computes $y_A = g^{x_A}$. Alice gives her long term public key $PK_A =< p, q, g, y_A >$ to the LEA, authenticates it by non-cryptographic (out-of-band) means, and gets a certificate $Cert(ID_A, PK_A)$. Then,

1 Alice chooses shares $x_1 \in_R Z_q^*$ and $x_2 = x_A(x_1)^{-1}$. Alice gives the shares x_1, x_2 privately to the escrow agents EA_1, EA_2, respectively.

2 The escrow agents check that $y_A = DH(g^{x_1}, g^{x_2})$. If not, Alice is reported to the LEA.

Key updating (session $t = 1, 2, \dots$). Agents EA_1, EA_2 choose numbers $r_{1,t}$, $r_{2,t} \in_R Z_r^*$ respectively, and jointly construct $h^{r_t} = h^{r_{1,t} r_{2,t}}$ in a secure way by using the Diffie-Hellman key exchange protocol [10]. The agents send h^{r_t} to the LEA which publishes it. This number identifies the session t, and is used by *all* the users of the system. It represents the randomness of the escrow agents in the key updating procedure and is the same for all users. The agents then discard the exponents $r_{1,t-1}$ and $r_{2,t-1}$ of the previous session (when $t > 1$). Then:

1 Alice chooses a number $r_{A,t} \in_R Z_r^*$, computes $h^{r_{A,t}}$ and sends this to the LEA. Alice also computes the Diffie-Hellman key $h_t = h^{r_t r_{A,t}}$.

2 Alice updates her secret key for session t to $SK_{A,t} = h_t x_A$. She then computes $y_{A,t} = g^{h_t x_A}$, and sends to the LEA her public

session key $PK_{A,t} = < p, q, r, g, h, y_{A,t} >$. Alice then proves in zero knowledge (see the Appendix) that $y_{A,t} = g^{DH(h^{r_t}, h^{r_{A,t}})DL(g^{x_A})}$, where $DL(g^{x_A})$ is the discrete logarithm of g^{x_A}. If the proof is correct, the LEA certifies the updated public key and issues Alice with a certificate $Cert(ID_A, PK_{A,t})$. Then Alice discards $r_{A,t}$ and the previous session key.

Getting an escrowed key. Assume that a court order has been issued to decrypt all ciphertexts intended for Alice during session t. Then the LEA will wiretap the communication of Alice. Let $(g^k, m(y_{A,t})^k)$ be an ElGamal encryption of a message m sent to Alice during this session. The LEA will send g^k and $h^{r_{A,t}}$ to the escrow agents. The agents first compute the Diffie-Hellman key $h_t = h^{r_t r_{A,t}}$, and then the factor $(y_{A,t})^k = (((g^k)^{h_t})^{x_1})^{x_2}$. They send $(y_{A,t})^k$ to the LEA for decryption.

Theorem 1 *If the Decision Diffie-Hellman problem is hard then the proposed escrow scheme has strong forward security.*

Proof. Suppose that there is a polynomial time algorithm $\mathcal{A}$ that breaks the proposed escrow scheme. Let $z, h^a, h^b \in_R Z_q^*$ be an input for the Decision Diffie-Hellman problem. We shall use $\mathcal{A}$ to break the DDH problem.

Choose at random a secret key $x_A \in_R Z_q^*$ and let $y_A = g^{x_A}$ be the long-term public key. Next, prepare a history of ciphertext-message pairs (c, m) for $\mathcal{A}$, for earlier sessions j, by choosing at random $k \in_R Z_q^*$, $m \in_R Z_p^*$ and $r_j, r_{A,j} \in_R Z_r^*$ and take $c = (g^k, mg^{kx_A h^{r_j r_{A,j}}})$.

Give to $\mathcal{A}$: x_A, y_A, and z, h^a, h^b instead of h_t, $h^{r_{A,t}}$, h^{r_t}, the public (session) key $y_{A,t} = g^{zx_A}$, a history of ciphertext-message pairs and the "ciphertext": (g^k, mg^{kzx_A}). Let the output of A be m'. If $m' = m$ then the decision is that $z = h^{ab}$, else $z \neq h^{ab}$.

Remark 3. The interactive zero knowledge proof in Step 2 of the key updating can be replaced by a signature, using the Fiat–Shamir heuristic [13], which requires a hash function. However it should be noted that if we use such signatures then the security of the scheme can only be proven in the Random Oracle Model [3].

Remark 4. In Section 2 we considered a solution involving the distribution of the secret keys via secret sharing in a proactive way. In our protocol above we also distribute the keys and use an updating mechanism similar to proactive mechanisms. However, our encryptions do not require a distributed computation.

Remark 5. The escrow agents are safe repositories for the long-term secret keys of all the users of the system. In our protocol the agents also generate a random number h^{r_t}. This number is for a specific time period and is the same for *all the users of the system*. In the next session a new random number is chosen and the old one is discarded. Observe that the addition or the removal of a user from the system does not affect the functionality of the agents.

Remark 6. The ElGamal escrow scheme described above can easily be modified to get a Key Recovery scheme by replacing the LEA and the escrow agents with a Data Recovery Agency and recovery agents respectively. Observe that if the keys to be recovered encrypt archived data, then there is no point in adopting a Key Recovery scheme with forward secrecy, as observed in [1]. Consequently, the proposed scheme can only be used to recover encrypted traffic.

Generalizations

1 It is easy to see how to generalize this scheme to a t-out-of-l key escrow scheme. Robustness can be achieved by using the approach in [16, 17]. Furthermore, our scheme can be easily modified to prevent subliminal channel attacks, as described in [24].

2 It is well known that the ElGamal encryption scheme is not semantically secure [18]. To extend our scheme to a semantically secure scheme we can use the Cramer-Shoup extension of ElGamal [7].

5. CONCLUSION

Forward security protects cryptographically processed data prior to key exposure. However in many applications it is difficult to detect intrusions. Indeed, hackers will not necessarily use the appropriated keys until this is expedient or profitable. It is therefore important to consider mechanisms, which also protect cryptographically processed data after an intrusion. Strong forward security offers such protection.

References

[1] H. Abelson, R. Anderson, S. Bellovin, J. Benaloh, M. Blaze, W. Diffie, J. Gilmore, P. Neumann, R. Rivest, J. Schiller, B. Schneier. *The Risks of Key Recovery, Key Escrow, Trusted Third Party & Encryption*, Digital Issues, No. 3, 1998, pp. 1–18.

[2] M. Bellare, S. Miner. *A Forward-Secure Digital Signature Scheme*, Advances in Cryptology — Crypto '99, Proceedings, Lecture Notes

in Computer Science, Vol. 1666, Springer 1999, pp. 197–207.

[3] M. Bellare, P. Rogaway. *Random Oracles are Practical*, 1st Annual Conference on Computer and Communications Security, Proceedings, ACM 1993, pp. 154–164.

[4] D. Boneh. *The Decision Diffie-Hellman Problem*, 3rd Algorithmic Number Theory Symposium, Proceedings, Lecture Notes in Computer Science, Vol. 1423, Springer 1998, pp. 48–63.

[5] M. Burmester, Y. Desmedt, J. Seberry. *Equitable Key Escrow with Limited Time Span (or How to Enforce Time Expiration Cryptographically*, Advances in Cryptology — AsiaCrypt '98, Proceedings, Lecture Notes in Computer Science Vol. 1514, Springer 1998, pp. 380–391.

[6] D. Chaum. *Zero-Knowledge Undeniable Signatures*, Advances in Cryptology — Eurocrypt '90, Proceedings, Lecture Notes in Computer Science, Vol. 473, Springer 1991, pp. 458–464.

[7] R. Cramer, V. Shoup. *A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack*, Advances in Cryptology — Crypto '98, Proceedings, Lecture Notes in Computer Science, Vol. 1462, Springer 1998, pp. 13–25.

[8] D.E. Denning and D.K Branstad. *A taxonomy of Key Escrow Encryption Systems*, Communications of the ACM, Vol. 39(3), 1996, pp. 24–40.

[9] Y. Desmedt, Y. Frankel. *Threshold Cryptosystems*, Advances in Cryptology — Crypto '89, Proceedings, Lecture Notes in Computer Science, Vol. 435, Springer 1989, pp. 307–315.

[10] W. Diffie, M. Hellman. *New Directions in Cryptography*, IEEE Transactions on Information Theory, Vol. 22(6), IEEE 1976, pp. 644–654.

[11] W. Diffie, P. Van Oorschot and M. Wiener. *Authentication and Authenticated Key Exchanges*, Designs, Codes and Cryptography, Vol. 2, 1992, pp. 107–125.

[12] T. ElGamal. *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory, Vol. 31, IEEE 1985, pp. 469–472.

[13] A. Fiat, A. Shamir. *How to prove yourself: Practical Solutions to Identification and Signature Problems*, Advances in Cryptology — Crypto '86, Proceedings, Lecture Notes in Computer Science, Vol. 263, Springer 1986, pp. 186–194.

[14] Y. Frankel, P. Gemmel, P.D. MacKenzie, and M. Yung. *Proactive RSA*, Advances in Cryptology — Crypto '97, Proceedings, Lecture Notes in Computer Science, Vol. 1109, Springer 1997, pp. 440–454.

[15] Y. Frankel, P. Gemmell, P. D. MacKenzie and M. Yung. *Optimal-Resilience Proactive Public-Key Cryptosystems*, 38th Annual Symp. on Foundations of Computer Science, Proceedings, IEEE 1997, pp. 384–393.

[16] Y. Frankel, P. Gemmel, and M. Yung. *Witness Based Cryptographic Program Checking and Robust Function Sharing*, 28th annual ACM Symp. Theory of Computing, Proceedings, ACM 1996, pp. 499–508.

[17] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. *Robust and Efficient Sharing of RSA Functions*, Advances in Cryptology — Crypto '96, Proceedings, Lecture Notes in Computer Science, Vol. 1109, Springer 1996, pp. 157–172.

[18] S. Goldwasser, S. Micali. *Probabilistic Encryption*, Journal of Computer and System Sciences, Vol. 28, 1984, pp. 270–299.

[19] S. Goldwasser, S. Micali, and C. Rackoff. *The Knowledge Complexity of Interactive Proof Systems* Proceedings of the 17th ACM Symposium on the Theory of Computing STOC, ACM Press, Providence, Rhode Island, U.S.A., 1985, pp. 291–304.

[20] C. Gunther. *An Identity-based Key-Exchange Protocol*, Advances in Cryptology — Eurocrypt '89, Proceedings, Lecture Notes in Computer Science, Vol. 434, Springer 1989, pp. 29–37.

[21] S. Haber and W. Storenetta. *How to Timestamp a Document*, Advances in Cryptology — Crypto '90, Proceedings, Lecture Notes in Computer Science, Vol. 537, Springer 1990.

[22] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung. *Proactive Public Key and Signature Schemes*, 4th Annual Conference on Computer and Communications Security, Proceedings, ACM 1997, pp. 100–110.

[23] A. Herzberg, S. Jarecki, S. Krawczyk, and M. Yung. *Proactive Secret Sharing*, Advances in Cryptology — Crypto '95, Proceedings, Lecture Notes in Computer Science, Vol. 963, Springer 1995, pp. 339–352.

[24] J. Kilian, T. Leighton. *Fair Cryptosystems, Revisited*, Advances in Cryptology — Eurocrypt '95, Proceedings, Lecture Notes in Computer Science, Vol. 963, Springer 1995, pp. 208–220.

[25] H. Krawczyk. *Simple Forward-Secure Signatures For Any Signature Scheme*, Procceedings of the 7th ACM Conference on Computer and Communications Security, ACM Press, 2000, pp. 108-115.

[26] U. Maurer, Y. Wolf. *Diffie-Hellman Oracles*, Advances in Cryptology — Crypto '96, Proceedings, Lecture Notes in Computer Science, Vol. 1109, Springer 1996, pp. 268–282.

[27] D. McGrew, A. Sherman. *Key Establishment in Large Dynamic Groups*, Manuscript, Submitted to IEEE Transactions on Software Engineering.

[28] A. Menezes, P. Van Oorschot, S. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1997.

[29] T. Rabin. *A Simplified Approach to Threshold and Proactive RSA*, Advances in Cryptology — Crypto '98, Proceedings, Lecture Notes in Computer Science, Vol. 1462, Springer 1997, pp. 89–104.

[30] A. Shamir. *How to Share a Secret*, Communications of the ACM, Vol. 22, ACM 1979, pp. 612–613.

Appendix

Let

$$\begin{aligned}\mathcal{L} \;=\; &\{(p,q,r,g,g^a,h^b,h^c,z) \mid p,q,r \text{ primes}, \; p=2q+1,\; q=2r+1,\\ &g \text{ a generator of } Z_p^*, h \text{ a generator of } Z_q^*, a \in Z_q^*, b,c \in Z_r^*, \text{and}\\ &z \in Z_p^* \text{ with } z = g^{ah^{bc}} \bmod p\}.\end{aligned}$$

An interactive zero-knowledge proof of membership in $\mathcal{L}$

Input: $x = (p,q,r,g,g^a,h^b,h^c,z)$

Repeat ℓ **times** $(\ell = \Theta(\log p))$:

1 The Prover chooses $k \in_R Z_q^*$, $t \in_R Z_r^*$, computes $u = ka \bmod q$, $v = c + t \bmod r$, and then sends to the Verifier:

$$X = g^{uh^{bv}}, \; Y = g^u, \; Z = h^v.$$

2 The Verifier sends to the Prover a bit query $e \in \{0,1\}$.

3 The Prover sends to the Verifier:

$$\begin{array}{ll} (u,v), & \text{if } \; e=0 \\ (k,t), & \text{if } \; e=1. \end{array}$$

Verification: The Verifier checks that:

$$\begin{array}{llll} \text{when } e=0, & X = g^{u(h^b)^v}, & Y = g^u, & Z = h^v \\ \text{when } e=1, & X = z^{k(h^b)^t}, & Y = (g^a)^k, & Z = h^c h^t. \end{array}$$

The Verifier accepts (that $x \in \mathcal{L}$) if the verification is satisfied for all k rounds.

Proof of correctness

Completeness: If $x \in \mathcal{L}$ then the Verifier will always accept.

Soundness: If the Verifier accepts with non-negligible probability $(\geq 1/poly(\log p))$, then the Prover must answer correctly both queries $e=0$, $e=1$ for some triple X,Y,Z. Therefore,

$$\begin{array}{lllll} Z & = & h^v = h^c h^t & \Rightarrow & v = c + t \bmod r \\ Y & = & g^u = (g^a)^k & \Rightarrow & u = ka \bmod q \\ X & = & g^{u(h^b)^v} = g^{kah^{b(c+t)}} = z^{k(h^b)^t} & \Rightarrow & z = g^{ah^{bc}}. \end{array}$$

It follows that $x \in \mathcal{L}$.

Simulation (zero-knowledge):

when $e = 0$, choose random u, v and construct X, Y, Z as in Step 1;
when $e = 1$, choose random k, t and construct $X = z^{k(h^b)^t}$, $Y = (g^a)^k$, and $Z = h^c h^t$.