Okamoto-Tanaka Revisited: Fully Authenticated Diffie-Hellman with Minimal Overhead*

Rosario Gennaro, Hugo Krawczyk, and Tal Rabin

IBM T.J. Watson Research Center
Hawthorne, New York 10532
`rosario@us.ibm.com, hugo@ee.technion.ac.il, talr@us.ibm.com`

Abstract. This paper investigates the question of whether a key agreement protocol with the same communication complexity as the original Diffie-Hellman protocol (DHP) (*two* messages with a *single* group element per message), and similar low computational overhead, can achieve forward secrecy against active attackers in a *provable* way. We answer this question in the affirmative by resorting to an old and elegant key agreement protocol: the Okamoto-Tanaka protocol [22]. We analyze a variant of the protocol (denoted mOT) which achieves the above goal. Moreover, due to the identity-based properties of mOT, even the sending of certificates (typical for authenticated DHPs) can be avoided in the protocol.

As additional contributions, we apply our analysis to prove the security of a recent multi-domain extension of the Okamoto-Tanaka protocol by Schridde et al. and show how to adapt mOT to the (non id-based) certificate-based setting.

1 Introduction

Since the invention of the Diffie-Hellman protocol (DHP) [10], much work has been dedicated to armor the protocol against active ("man in the middle") attacks. Designing *authenticated Diffie-Hellman protocols* has proven to be very challenging at the design and analysis level, especially when trying to optimize performance (both computation and communication). This line of work has been important not only from the practical point of view but also for understandings what are the *essential limits* for providing authentication to the DHP.

In particular, it has been shown that one can obtain an *authenticated* DH protocol with the same communication as the basic unauthenticated DHP (at least if one ignores the transmission of public key certificates); namely, a 2-message exchange where each party sends a single DH value, and where the two messages can be sent in any order. A prominent example of such protocols is MQV [18] (and its provably-secure variant HMQV [17]) where the cost of computing a session key is as in the basic unauthenticated DHP plus half the cost of one exponentiation (i.e., one off-line exponentiation and 1.5 on-line exponentiations).

Protocols such as the 2-message MQV are "implicitly-authenticated protocols;" that is, the information transmitted between the parties is computed

* Extended Abstract. Full version available at http://eprint.iacr.org/2010/068.

without access to the parties' long-term secrets while the authentication is accomplished via the computation of the session key that involves the long-term private/public keys of the parties. Unfortunately, implicitly-authenticated protocols, while offering superb performance, are inherently limited in their security against active attackers. Indeed, as shown in [17], such protocols can achieve perfect forward secrecy (PFS) *against passive attackers only.*

Recall that PFS ensures that once a session key derived from a Diffie-Hellman value is erased from memory, there is no way to recover the session key even by an attacker that gains access to the long-term authentication keys of the parties after the session is established. PFS is a major security feature that sets DHPs apart from other key agreement protocols (such as those based in PK encryption) and is the main reason for the extensive use of DHPs in practice (e.g., IPsec and SSH). Adding PFS against active attackers to protocols like MQV requires increased communication in the form of additional messages and/or explicit signatures.

In this paper we investigate the theoretical and practical question of whether the limits of DHPs can be pushed further and obtain a protocol with full security against active attackers (including PFS) while preserving the communication complexity of a basic DHP (*two* messages with a *single* group element per message) and low computational overhead. We answer this question in the affirmative by departing from implicitly authenticated protocols and resorting to an old and elegant key agreement protocol: the Okamoto-Tanaka protocol [22]. We analyze a variant of the protocol (denoted mOT) which achieves the above minimal communication, incurs a negligible computational overhead relative to a basic DHP over an RSA group, and yet achieves *provable* security including *full PFS* against active attackers[1]. Moreover, due to the identity-based [24] properties of mOT, even the sending and verification of certificates is avoided in the protocol.

Our Results. The protocol mOT we analyze is a "stripped down" version of the "affiliation-hiding" key exchange protocol by Jarecki *et al.* [15] (a version of key agreement where parties want to hide who certified their public keys). We remove all the extra steps designed to obtain affiliation-hiding (which is not a concern for our paper) and focus on the 2-message version of the [15] protocol (the latter includes a third message and the transmission of additional authentication information for the parties to confirm they indeed have the same key). We present a rigorous proof of security for mOT in the Canetti-Krawczyk (CK) Key-Agreement Protocol model [5]. The security of the protocol in this model, including weak PFS (i.e., against passive attacks only), is proven in the random oracle model under *the standard RSA assumption.* For the proof of *full* PFS against active attackers (and only for this proof) we resort to non-black-box assumptions in the form of the "knowledge of exponent" assumptions. We stress that our goal is to prove full security (including full PFS) for the 2-message protocol *without* the extra key

[1] There are DH protocols that provide full PFS against active attacks with just two messages, but they require to send (and process) additional information, e.g. explicit signatures [27] or encrypted challenges [16].

confirmation steps: indeed the 3-message protocol with key confirmation can be proven secure (including full PFS) under the standard RSA assumption without requiring extra assumptions (this proof is actually implicit in [15]).

Modified Okamoto-Tanaka (mOT). The modified Okamoto-Tanaka protocol mOT, is described in Figure 1 (for a precise specification see Section 3). We describe the protocol as an identity-based protocol using a KGC (key generation center) as this setting provides added performance advantages to the protocol. Following [15], we include hashing operation on identities as well as the hashing and squaring operations in the computation of the session key K (these steps are not part of the original Okamoto-Tanaka).

The Modified Okamoto-Tanaka (mOT) Protocol

Setting: A Key Generation Center (KGC) chooses RSA parameters $N = pq$ (such that p and q are random safe primes), and exponents d, e, and a random generator g of QR_N, the (cyclic) subgroup of quadratic residues $\bmod N$.

KGC publishes N, e, g, two hash functions H (with range QR_N) and H' (with range of the desired length of the session key), and distributes to each user U with identity id_U a private key $S_U = H(id_U)^d \bmod N$.

Key agreement: A and B choose ephemeral private exponents x and y, respectively.

$$\begin{tikzcd} A \arrow[r, "\alpha = g^x S_A \bmod N"] & B \arrow[l, bend left, "\beta = g^y S_B \bmod N"] \end{tikzcd}$$

$$\bar{K}_A = (\beta^e / H(id_B))^{2x} \bmod N \qquad \bar{K}_B = (\alpha^e / H(id_A))^{2y} \bmod N$$

$$\bar{K} = \bar{K}_A = \bar{K}_B = g^{2xye} \bmod N$$

A and B set the session key to $K = H'(\bar{K}, id_A, id_B, \alpha, \beta)$

Fig. 1. A and B share session key K. See Section 3 for full details.

Security Proof and Full PFS "for free". The security result that sets our protocol and work apart is our proof of *full PFS* for mOT, namely, *perfect forward secrecy against fully active attackers.* The proof of full PFS (and only this proof) requires two additional "non-black-box" assumptions: one is the well-known KEA1 (knowledge of exponent) assumption [8,1] related to the hardness of the Diffie-Hellman problem and the second is similar in spirit but applies to the discrete logarithm problem (see Section 4). Enjoying full PFS is a major advantage of mOT relative to efficient two-message protocols such as MQV that can only offer weak PFS. Indeed, in spite of mOT transmitting a single group element in each of the two messages, it overcomes the inherent PFS limitations of implicitly authenticated DHPs by involving the sender's private key in the computation of each protocol's message. Most importantly, as we explain below, this *full security against active attackers* is achieved with zero communication and negligible computational overhead relative to the basic DHP. We believe this to be not just a practical feature of mOT but also a significant contribution to the theory of key agreement protocols showing that armoring the original DHP against active attackers can be achieved essentially "for free".

Performance. The cost of mOT remains essentially the same as in the basic (unauthenticated) DHP: one message per party, that can be sent in any order, with each message containing a single group element. No additional authentication information needs to be transmitted. Thanks to the identity-based properties of the protocol, public-key certificates need not be sent or verified. The only extra operation is one exponentiation to the e-th power, which can be chosen to be 3, and one squaring; that is, just three modular multiplications in all. However, note that mOT works over an RSA group and therefore exponentiations are more expensive than over elliptic curves (where protocols like MQV can be run). Yet, we also note that mOT can be implemented with short exponents, say 160-bit exponents when the modulus is of size 1024 (or a 224-bit exponent with a 2048-bit modulus). Our proof of the protocol holds in this case under the common assumption that in the RSA group the discrete logarithm problem remains hard also for these exponent sizes. In terms of practical efficiency, for moderate security parameters (160-200 bit exponents) the cost of one on-line exponentiation in mOT is competitive with the 1.5 exponentiations over elliptic curves required by MQV. For larger security parameters the advantage is fully on the elliptic curve side though in this case one has to also consider the overhead incurred by certificate processing in a protocol like MQV (which is costly especially for ECDSA-signed certificates).

Of course, beyond the practical performance considerations, mOT holds a significant security advantage over 2-message MQV, namely, its full PFS against active attackers. The fact that mOT can do so well with almost no overhead over the underlying basic Diffie-Hellman protocol, and with full security against active attacks, is an important theoretical (and conceptual) aspect of our work pointing to the limits of what is possible in this area.

More discussion on the performance mOT can be found in the full version.

The Need for a Key Generation Center (KGC). As an identity-based protocol, mOT avoids the need for certificates (a significant communication and computational advantage). The id-based setting, however, introduces the need for a KGC that generates and distributes keys to users. This results in a different trust model than the traditional certification authority (CA) that certifies public keys but does not generate or know the private keys of parties. Note, however, that in mOT the private keys are used only for authentication. Thus, while a KGC can impersonate a party, it cannot learn keys exchanged by that party (we note – see full version – that the PFS property holds also against a corrupted KGC). Note that a regular CA can also impersonate parties at will by issuing certificates with the user's name but with a private key known to the CA. Interestingly, as we show below, mOT can be modified to work also in the traditional CA setting.

Further results. In the full version we extend the above proofs and analysis to a recent extension of the Okamoto-Tanaka protocol proposed by Schridde, Smith and Freisleben [25] that allows the execution of the protocol between users that belong to different domains, i.e., to different key generation centers (KGC).

The full version also shows that the mOT protocol and its extension from [25] can be modified to work as "traditional" (i.e., not ID-based) key agreement protocol.

Related work. Key agreement protocols (KAPs) have played an important role in the development of identity-based cryptography, with Okamoto [21], Okamoto and Tanaka [22], Gunther [12] being early examples of id-based cryptography. (Even earlier, the work by Blom [2] on key distribution can be seen as a precursor of id-based schemes.) With the flourishing of pairings-based cryptography, many more id-based KAPs have been designed; yet getting them right has been a challenging task. See the survey by Boyd and Choo [3] and Chen, Cheng, and Smart [6] for good descriptions and accounts of the main properties of many of these protocols. Even to date it seems that very few (e.g., [4,29]) were given full proofs of security (many others were broken or enjoy only a restricted notion of security, such as partial resistance to known-key attacks). In all, the mOT protocol studied here compares very favorably with other id-based and traditional KAPs in provability and security properties (e.g., PFS) as well as performance-wise.

We already discussed the relationship of our work with [15]. We stress again that the security analysis there is for the protocol with the extra key confirmation messages, while we analyze the minimalistic 2-message protocol in Figure 1.

Multi-domain extensions of id-based KAPs have been proposed in [7,19] but without full proofs of security. The multi-domain extension of the mOT protocol that we fully analyze here is from Schridde et al. [25] which also contains a good discussion of the benefits of multi-domain identity-based protocols.

In general, while interactive authenticated KAPs, especially those authenticated with signatures, can easily accommodate certificates (which a party can send together with its signature), avoiding the need for certificates constitutes a significant practical simplification of many systems. In particular, they provide more convenient solutions for revocation and less management burden [28]. The Okamoto-Tanaka example shows that the identity-based setting can sometimes even improve performance.

Open questions. We believe that the mOT protocol is remarkable for its "minimalism", providing full and provable authenticated key-agreement security (including full PFS) with the same communication and minimal computational overhead relative to the underlying unauthenticated DHP. Yet there are several ways one could hope to improve on this protocol and on our results;

achieving any of these improvements would bring us even closer to the "ultimate" authenticated DHP:

(i) Find a protocol with the same communication/computation/security characteristics as mOT but which works over arbitrary dlog groups (in particular elliptic curves). In this case, the minimalism of mOT would translate into optimal practical performance (even a certificate-based protocol with these properties would be very useful). (ii) Prove the full PFS security of mOT without resorting to non-black-box assumptions (while we believe that proofs under these assumptions carry a very strong evidence of security, using more standard assumptions is obviously desirable). (iii) Improve on mOT by avoiding the vulnerability of the protocol to the exposure of the DH values g^x, g^y or the ephemeral exponents x, y.

2 Preliminaries

Let $SPRIMES(n)$ be the set of n-bit long safe primes. Recall that a prime p is safe if $\frac{p-1}{2}$ is prime. Let $N = pq$ be the product of two random primes in $SPRIMES(n)$; denote $p = 2p' + 1$ and $q = 2q' + 1$. Let e be an integer which is relatively prime to $\phi(N) = 4p'q'$.

We say that the RSA Assumption (with exponent e) holds if for any probabilistic polynomial time adversary $\mathcal{A}$ the probability that $\mathcal{A}$ on input N, e, R, where $R \in_R Z_N^*$, outputs x such that $x^e = R \bmod N$ is negligible in n. The probability of success of $\mathcal{A}$ is taken over the random choices of p, q, R and the coin tosses of $\mathcal{A}$.

Remark (semi-safe primes). For simplicity, we assume that $p = 2p' + 1$ and $q = 2q' + 1$ are safe primes, namely, p' and q' are prime. We can relax this assumption to require that $\gcd(p', q') = 1$ and that neither p' or q' have a prime factor smaller than 2^ℓ for a given security parameter ℓ. With these assumptions we get that QR_N is cyclic and that a random element in QR_N is a generator with overwhelming $(1 - 2^{-\ell})$ probability, two properties that we use in our construction and analysis.

Throughout the paper we use the following well-known result of Shamir [23]:

Lemma 1. *Let N, e, d be RSA parameters and f be an integer relatively prime to e. There is an efficient procedure that given N, e, f (but not d) and a value $(x^f)^d \bmod N$, for $x \in Z_N^*$, computes $x^d \bmod N$.*

The cyclic group QR_N. If N is an RSA modulus product of safe primes, then the subgroup QR_N of quadratic residues in Z_N^* is cyclic of order $p'q'$. Let g be a random generator of QR_N (such generator can be found by squaring a random element in Z_N^* (this algorithm yields a generator with overwhelming probability and the resulting distribution is statistically close to uniform). In protocols and proofs below we are going to generate random elements in the group generated by g according to the uniform distribution and with known exponents (i.e., their dlog to the base g). Such a random element X could be generated by choosing an integer

$x \in [1..p'q']$ uniformly at random and setting $X = g^x \bmod N$. But this option implies knowledge of the factorization of N (knowing the value $p'q'$ is equivalent to factoring N). Parties who do not know the value $p'q'$ can approximate the uniform distribution over $\langle g \rangle$ as follows: generate $x \in [1..\lfloor N/4 \rfloor]$ and set $X = g^x \bmod N$. It is not hard to see (cf. [9]) that if p' and q' are of the same size (as required here) then the uniform distributions over $[1..p'q']$ and $[1..\lfloor N/4 \rfloor]$ are statistically close with an exponentially small gap.

Let g be a random generator of QR_N and let $X = g^x \bmod N$ and $Y = g^y \bmod N$ two random elements in QR_N. We say that the Computational Diffie-Hellman (CDH) Assumption (for N and g) holds if for any probabilistic polynomial time adversary $\mathcal{A}$ the probability that $\mathcal{A}$ on input N, g, X, Y outputs Z such that $Z = g^{xy} \bmod N$ is negligible in n. The probability of success of $\mathcal{A}$ is taken over the random choices of p, q, x, y and the coin tosses of $\mathcal{A}$. We know from [26] that the hardness of factoring N (and therefore the RSA Assumption) implies the CDH Assumption.

3 The Modified Okamoto-Tanaka Protocol

Protocol Setup. A key generation center KGC (for "trusted authority") chooses an RSA key (N, e, d), where N is the product of two safe primes p, q. As usual e, d are such that $ed = 1 \bmod \phi(N)$. The KGC also chooses a random generator g for the subgroup of quadratic residues QR_N. The public key of the KGC is (N, e, g) and its secret key is d.

Two hash functions H, H' are public parameter. The first function H outputs elements in the group generated by g (this can be achieved by setting H to be the square $\bmod N$ of another hash function with range Z_N^*). The second hash function H' outputs k-bit strings, where k is the length of the required session key.

Each user in the system has an identity; for convenience we sometimes associate a name to an identity. For example, Alice will be the name of a party while her identity is denoted id_A. We also denote $A = H(id_A)$ and $B = H(id_B)$ (thus, A, B are elements in QR_N). When Alice requests her secret key from the KGC, she receives the value $S_A = A^d \bmod N$ as her secret key (we can think of this as the KGC's RSA signature on Alice's id).

The key agreement. Alice chooses a random integer x from a set S that we specify below. She then computes $X = g^x \bmod N$ and sends to Bob $\alpha = X \cdot S_A \bmod N$. Bob chooses a random y in S, sets $Y = g^y \bmod N$, and sends to Alice the value $\beta = Y \cdot S_B \bmod N$. Alice computes a shared secret value $\bar{K}_A = (\beta^e/B)^{2x} \bmod N$ while Bob computes it as $\bar{K}_B = (\alpha^e/A)^{2y} \bmod N$. (Alice and Bob check that the incoming value is in Z_N^* or else they abort the session.)

Notice that the values $\bar{K}_A, \bar{K}_B$ are equal:

$$\bar{K}_A = Y^{2xe} \cdot (S_B^e/B)^{2x} = Y^{2xe}(B^{ed}/B)^{2x} = g^{2xye} = ((XS_A)^e/A)^{2y} = \bar{K}_B$$

Alice and Bob set $\bar{K}$ as this shared secret value $\bar{K} = \bar{K}_A = \bar{K}_B = g^{2xye} \bmod N$ and set the session key to $K = H'[\bar{K}, id_A, id_B, \alpha, \beta]$. Since we want both parties to compute the same session key we need to determine an ordering between id_A and id_B, and between α and β in the input to the hash; for example, a lexicographic ordering (note that we are not assuming necessarily that there is a definite role of "initiator" and "responder" in the protocol, and hence we do not use such roles to determine the ordering of the above values).

Protection of ephemeral values. We specify that the ephemeral Diffie-Hellman values X, Y chosen by the parties be given the same protection level as the private keys S_A, S_B (indeed, learning these ephemeral values is equivalent to learning the private keys). In particular, if these values are stored in the (less secure) session state they need to be stored encrypted under a (possibly symmetric) key stored with the private key. (This is analogous to the need for protecting the ephemeral value k in a DSA signature.) In addition, we specify that in the computation of the session key, the hashing of the value $\bar{K}$ be performed in protected memory and only the session key be exported to a session state or application (learning the shared secret $\bar{K}$ value opens some attack venues as explained in the full version).

The exponents set S and performance considerations. The performance cost of the protocol is dominated by the exponentiation operations; hence the choice of the set S from which ephemeral exponents are selected is important. A first choice would be to define S as the interval $[1..\lfloor\sqrt{N}/2\rfloor]$. In this case, as shown in [11] (following the results in [14]), the two distributions

$$\{g^x \text{ for } x \in_R [1..\lfloor N/4\rfloor]\} \quad \text{and} \quad \{g^x \text{ for } x \in_R [1..\lfloor\sqrt{N}/2\rfloor]\}$$

are computationally indistinguishable under the assumption that factoring N is hard. Therefore, one can use exponents of length *half the modulus* without any loss in security. However, performance can be significantly improved by setting S to be the set of exponents of length κ, where κ is twice the security parameter (e.g., $\kappa = 224$). Indeed in this case, the security of the protocol relies on the common assumption that discrete log (over Z_N^*) is hard also when the exponents are of length κ. Indeed (see Lemma 3.6 in [11]), this assumption implies that the two distributions

$$\{g^x \text{ for } x \in_R [1..\lfloor N/4\rfloor]\} \quad \text{and} \quad \{g^x \text{ for } x \in_R [1..2^\kappa]\}$$

are computationally indistinguishable, and hence using the short or long exponents is equivalent. Therefore, we recommend the protocol to be implemented using short exponents; in particular, we use this case when discussing the protocol's performance.

3.1 Proof of the mOT Protocol

We prove first the following theorem showing the basic security of the mOT protocol. In the full paper we prove further security properties, namely, resistance to KCI and to reflection attacks, and weak PFS. We defer the proof of full PFS (which requires a more involved proof and additional assumptions) to Section 4.

Theorem 1. *Under the RSA assumption, if we model H, H' as random oracles, the* mOT *protocol is a secure identity-based key agreement protocol.*

Proof. The proof is carried in the Canetti-Krawczyk Key-Agreement security model [5] (a succinct summary of this model is presented in the full version) and it follows a typical simulation/reduction argument: We assume an efficient KA-attacker $\mathcal{M}$ that breaks the security of the mOT protocol and use it to build an algorithm that inverts RSA on random inputs.

We start by noting the following fact about an attacker against mOT: Since the session identifier $(id_A, id_B, \alpha, \beta)$ is hashed together with the shared secret value $\bar{K}$ to obtain the session key K, we know that two different sessions necessarily correspond to two different session keys. Moreover, since the hash function H' is modelled as a random oracle then the only way for the attacker to calculate, identify, or distinguish a session key is by computing the value $\bar{K}$ and explicitly querying it from H'.[2]

We call the algorithm that we build for inverting RSA a "simulator" (denoted SIM) since it works by simulating a run of the mOT protocol against the KA-attacker $\mathcal{M}$ which is assumed to win the test-session game with non-negligible probability.

Input to SIM. The input to SIM is a triple (N, e, R) where N, e are chosen with the same RSA distribution as used in the mOT protocol and R is a random element in Z_N^*. The goal of SIM is to output $R^d \bmod N$ where d is such that $ed = 1 \bmod \phi(N)$.

Some conventions: We often omit the notation "mod$N$" when operating in the group Z_N^*. When saying that we chose a random element u from QR_N we mean choosing $v \in_R Z_N^*$ and setting $u = v^2 \bmod N$. To choose a value u in Z_N^* with a known "RSA signature" $s = u^d$, we first choose s and then set $u = s^e$. If $U = g^u, W = g^w$ are elements of QR_N we denote with $DH_g(U, W)$ the value g^{uw}, i.e. the result of the Diffie-Hellman transform in base g applied to U, W.

SIM runs a virtual execution of the mOT protocol (consisting of multiple sessions) against the attacker $\mathcal{M}$, simulating all protocol actions, including the determination of private keys and responses to queries to the functions H, H' made by $\mathcal{M}$. In particular, we allow SIM to "program" the hash functions H, H' (as long as outputs are chosen independently of each other and with uniform distribution) as is customary when modeling H, H' as random oracles.

Identities and keys. Each participant in the protocol has an identity, id_P, possibly chosen by $\mathcal{M}$; we denote $P = H(id_P)$. Of all party identities participating in the protocol, SIM chooses one at random; we denote it by id_B and will refer to this party as Bob. For each participant id_P *other than Bob*, SIM chooses a

[2] Note that if parties A and B have a session where they exchanged messages α (from A to B) and β (from B to A), and another session where the same messages were exchanged but in the reverse direction, both sessions will have the same key. However, as long as one of A and B is honest, each session will have at least one fresh message (except for the negligible probability that two random values in QR_N coincide), hence the above cannot happen even with the attacker's intervention (who can only choose one message in each session).

random value $p \in_R QR_N$ and sets $P = H(id_P) = p^e$. In this way, SIM also knows the private key of the participant, i.e., $S_P = P^d = p$ (note that P and S_P are elements of QR_N). For Bob, SIM sets $H(id_B) = B = R^2 \bmod N$, where R is the input to SIM (note that R^2 is random in QR_N).

Choosing a QR_N generator. SIM sets the random generator g of QR_N to be used in the protocol as following: it chooses random $\bar{r} \in_R QR_N$, sets $r = \bar{r}^e$, and $g = (rB)^e$. Note that with these choices $B = g^d/r$ and $\bar{r} = r^d$; also note that g and B are random in QR_N and independent.

Guessed test session. Before starting the simulation of session establishments, SIM chooses at random a (future) session that it conjectures will be chosen by $\mathcal{M}$ as the test session. SIM does so by guessing the holder of the test session among all the parties in the orchestrated protocol run (we refer to this party as Alice) and guessing the order number of the session among all of Alice's sessions. This allows SIM to know when the guessed session is activated at Alice in the protocol's run. In addition, SIM also guesses that the peer to the test session will be Bob (defined above). We specify that, if at any point in SIM's simulation, it is determined from the protocol's run that the guessed session is not to be chosen as the real test session by $\mathcal{M}$ (e.g., if either Alice or Bob are corrupted, or another test session is chosen by $\mathcal{M}$, etc.) the simulator aborts. The probability that the guessed session will actually be chosen by $\mathcal{M}$ as the test session is non-negligible (as long as the simulation of the protocol by SIM is correct).

Session Interactions (non-test sessions). Attacker $\mathcal{M}$ can choose to initiate and schedule sessions between any two participants and can input its own values into the various sessions, either by utilizing corrupted players or by delivering messages allegedly coming from honest parties. The simulator SIM needs to act on behalf of honest parties in these interactions. Simulating the actions of any uncorrupted party other than Bob is simple for SIM, as it knows their private keys and can choose their ephemeral exponents. Sessions in which Bob is a participant are more problematic since SIM does not know Bob's private key $S_B = B^d$. Whenever Bob is activated in a session, SIM will set the value $\beta = g^b/\bar{r}$ as the outgoing message from Bob where $b \in_R [1..\lfloor N/4 \rfloor]$ is chosen afresh with each activation of Bob and the value $\bar{r}$ is fixed and defined above. Clearly, β is distributed uniformly over QR_N as in the real runs of mOT. While SIM cannot compute session keys with such choice of β we will still see that it can answer the attacker's session-key queries.

Response to party corruption and session key queries (non-test sessions). If at any point $\mathcal{M}$ corrupts a party, SIM provides all information for that party including the private key (which SIM knows). Note that if the attacker asks to corrupt Bob, SIM aborts since it is a sign that SIM did not guess correctly the test session. Session-key queries for sessions where one of the messages was generated by an honest party other than Bob, can be answered by SIM who chooses the ephemeral exponent for the session. The problematic cases are sessions where Bob is a peer and for which the incoming message to Bob was chosen by the attacker (rather than by SIM itself), and provided to Bob as coming from some party id_C (we refer to it as Charlie), which may be honest or corrupted, but different than Bob[3]. In this case SIM does not know the ephemeral exponents of either party to the session so it cannot compute the session key. Instead the simulation proceeds as follows.

The idea is that as long as $\mathcal{M}$ does not query the session value $\bar{K}$ from the random oracle H', then SIM can answer the session key query with a random value. However, if $\mathcal{M}$ does know the value $\bar{K}$, and it actually queries H' on this value, then SIM needs to answer consistently. Specifically, we are dealing with a session where the peers are Bob and Charlie, whose hashed identities are $B = H(id_B)$ and $C = H(id_C)$, respectively, and the exchanged values are γ, chosen by the attacker, and β chosen by SIM as specified above. Thus, the session key is $H'(\bar{K}, id_C, id_B, \gamma, \beta)$ for the appropriately computed $\bar{K}$. Before answering the session-key query, SIM needs to check whether an input of the form $(Q, id_C, id_B, \gamma, \beta)$ was queried from H' where $Q = \bar{K}$. If such a query with $Q = \bar{K}$ was indeed performed then SIM will answer the session-key query with the existing value $H'(Q, id_C, id_B, \gamma, \beta)$. If not, SIM will choose a random value ρ in the range of H' and will return ρ as the value of the session key.

The main question is how will SIM verify whether $Q = \bar{K}$ for a prior query. The value $\bar{K}$ can be represented as $DH_g(Z^2, \beta^e/B)$ for $Z = \gamma/S_C$. By our choice of $B = g^d/r$, $\beta = g^b/\bar{r}$ and $r = \bar{r}^e$, we have that $\beta^e/B = (g^{eb}\bar{r}^{-e})/(g^d r^{-1}) = g^{eb-d}$ and therefore,

$$\bar{K} = DH_g(Z^2, \beta^e/B) = DH_g(Z^2, g^{eb-d}) = Z^{2(eb-d)} \tag{1}$$

Now, since exponentiation to the e is a permutation over Z_N^*, we have that $Q = \bar{K}$ if and only if $Q^e = \bar{K}^e$, and by Equation (1) this is the case if and only if $Q^e = (Z^{2(eb-d)})^e = Z^{2(e^2b-1)}$. But this last computation can be performed by SIM who knows all the involved values, including b that SIM chose and Z (since $Z = \gamma/S_C$ and SIM knows both γ and S_C).[4]

Simulating the test session. When $\mathcal{M}$ activates the session at Alice that SIM chose as its guess for the test session, SIM acts as follows. Let the identity of Alice be id_A and denote $A = H(id_A)$. Since Alice is assumed to be the holder of the test session, it means that it is Alice (or SIM in our case) who chooses the outgoing message α from the session, not the attacker. SIM sets this message to the value $\alpha = (rB)^f S_A$, where r, B are as described at the begining of the simulation, $S_A = A^d$ is Alice's private key (which SIM knows) and f is chosen as $f = te+1$ for $t \in_R [1..\lfloor N/4 \rfloor]$. With this choice, α's distribution is statistically close to uniform over QR_N. Indeed, we have $(rB)^{te+1} = (g^d)^{te+1} = g^d g^t$ with

[3] We assume for the time being that Bob does not run a KA session with itself (thus $C \neq B$). The case where both session peers have the same identity is called a reflection attack and is proved in the full version.

[4] We note for future reference, that knowing S_C is not strictly necessary for SIM to carry this simulation step. If SIM does not know S_C (as in some other proofs in this paper) it does not know Z either. Instead SIM will use $Z^e = \gamma^e/C$ which it does know, and instead of checking $Q^e = Z^{2(e^2b-1)}$ it will check the equivalent $Q^{e^2} = (Z^e)^{2(e^2b-1)}$.

$t \in_R [1..\lfloor N/4 \rfloor]$ (recall that in the real protocol Alice chooses $\alpha = g^x S_A$ with g^x also statistically close to uniform distribution over QR_N). It also makes it independent of other values in the protocol including B and β.

The peer to the test session is Bob (or else SIM aborts) and the incoming message is denoted by β. This value can be chosen by the attacker (which delivers it to Alice as coming from Bob) or by Bob itself. In the latter case, β is chosen by SIM as described above for other sessions activated at Bob. In case $\mathcal{M}$ chooses β, it can be any arbitrary value. Below, we make no assumption on β other than being in Z_N^*. The session key in this case is $K = H'(\bar{K}, id_A, id_B, \alpha, \beta)$ where $\bar{K}$ is computed as follows: if $X = g^x$ denotes the value α/S_A then $\bar{K} = (\beta^e/B)^{2x}$. Now, by our choice of parameters α, g, B, r (in particular, $rB = g^d$), we have that $X = \alpha/S_A = (rB)^f = (g^d)^f$ and hence $x = df \bmod \phi(N)/4$. Thus,

$$\bar{K} = (\beta^e/B)^{2df} \tag{2}$$

Since SIM cannot compute this value (it does not know the ephemeral exponent of either peer to the session) we need to show how SIM responds to a test-session query (assuming the guessed session is indeed chosen by $\mathcal{M}$ as the test session). Upon such a query, SIM will check if there was any query made to H of the form $(Q, id_A, id_B, \alpha, \beta)$ and if so, it will check if Q equals the session value $\bar{K}$. This is done by checking whether $Q^e = (\beta^e/B)^{2f}$ (which involves values known to SIM). Indeed, note that $Q = \bar{K}$ if and only if $Q^e = \bar{K}^e$ (as exponentiation to e is a permutation) and using Equation (2) we have $\bar{K}^e = (\beta^e/B)^{2f}$. If SIM identifies such a Q, SIM has learned $\bar{K}$ from which it can compute its target RSA forgery as we explain below. If not, SIM responds to the session-key query with a random value. From now on, it monitors $\mathcal{M}$'s queries to H to see if a $Q = \bar{K}$ is identified, in which case SIM learns the session key and outputs the forgery.

Computing the forgery R^d. The goal of SIM is to compute $R^d \bmod N$ where $R \in_R Z_N^*$ was given to SIM as input. We now show that whenever SIM learns the session key corresponding to the test session (as shown above) it can compute R^d. Indeed, it is easy to see from Equation (2) that $(B^{2f})^d = \beta^{2f}/\bar{K}$, and since SIM chose $B = R^2$ then $(R^{4f})^d = \beta^{2f}/\bar{K}$. Using Lemma 1 and the fact that $4f$ is relatively prime to e we derive R^d from $(R^{4f})^d$.

Finally, we note that in order to win the test-session game with non-negligible advantage it must be that $\mathcal{M}$ queries the correct $\bar{K}$ from H with non-negligible probability, then SIM is guaranteed to learn $\bar{K}$, and hence compute R^d, also with non-negligible probability.

In the full version we prove further security properties of mOT , such as resistance to *reflection* and *key compromise* attacks.

4 Proof of the PFS Property of the mOT Protocol

In this section we prove that the Modified Okamoto-Tanaka protocol enjoys full *Perfect Forward Secrecy (PFS)* against active attackers. For this proof we need to resort to two additional assumptions[5] (on top of the RSA Assumption required for the proof of the basic security of the protocol, i.e. Theorem 1). The first assumption is the well-known "Knowledge of Exponent Assumption" introduced by Damgard [8] (and further used and studied in [13,1]). Intuitively, it states that to compute a DH value g^{xy} out of a triple g, g^x, g^y one has to necessarily know either x or y. We will refer to this assumption (called KEA1 in [1]) as KEA-DH.

Knowledge of Exponent Assumption for Diffie-Hellman (KEA-DH). Let G be a cyclic group, and let g, h be distinct generators of G. The assumption says that for every algorithm $\mathcal{M}$ that on input G, g, h outputs (y, z) where $y = g^x$ and $z = h^x$ for some integer x, there exists an algorithm $\mathcal{M}^*$ which outputs x.

A fully formal statement of the assumption can be found in [1]; in particular, it is assumed that for every set of random coins used by $\mathcal{M}$, if $\mathcal{M}$ outputs $(y = g^x, z = h^x)$ then for the same set of random coins $\mathcal{M}^*$ outputs x.

Our second assumption is close in spirit to KEA-DH but it applies to the discrete logarithm problem; we refer to it as KEA-DL*. The idea is as follows: Under the discrete log assumption, given a pair $(g, B = g^b)$ (for random b) it is hard to find b. But what if in addition to the pair $(g, B = g^b)$ one is also given a dlog oracle where one can input any value in G other than B, and receive its dlog to base g. Obviously, one can find b by querying, for example, the value Bg; more generally, one can query BV where $V = B^i g^j$ for known i, j, and compute b out of the dlog of BV. The KEA-DL* assumption states that if one is allowed a single query to the oracle then the above strategy is the only feasible one. Namely, if an algorithm finds b by querying a single value from the oracle then there is another algorithm that outputs values i, j as above. In addition we also need to assume that the knowledge of the e-th root of B (for a fixed value e) does not help the attacker to find the dlog of B in the above game. More formally[6]:

Modified Knowledge of Exponent Assumption for Discrete Log (KEA-DL*). Let G be the subgroup of Quadratic Residues in Z_N^* where N is an RSA modulus. We modify the **KEA-DL** assumption to allow $\mathcal{M}$ to receive also the e-root of B (where e is an RSA exponent). The modified assumption is as follows:

1. $Chall_{DL^*}$ provides $\mathcal{M}$ with $N, e, g, B = g^b$ where g, B are random quadratic residues in Z_N^*;
2. $\mathcal{M}$ is allowed to query an element $V \in G$;
3. $Chall_{DL^*}$ responds with the discrete log of BV and the e-root of B;
4. $\mathcal{M}$ outputs an integer b'; $\mathcal{M}$ wins if $b = b'$.

[5] It is important to note that, as shown in the full version, weak PFS – i.e. against passive attackers – is a direct consequence of Theorem 1 and does not require additional assumptions.

[6] A fully formal statement of the assumption quantifies over each set of random coins of algorithms $\mathcal{M}$ and $\mathcal{M}*$; the details are similar to the treatment of the KEA-DL* assumption in [1] and are omitted from this extended abstract.

The KEA-DL* assumption states that for every algorithm $\mathcal{M}$ that wins the above game, there exists an algorithm $\mathcal{M}^*$ which outputs integers i, j such that $V = B^i g^j$.

Theorem 2. *Under the RSA,* **KEA-DH** *and* **KEA-DL*** *assumptions, if we model H, H' as random oracles, then the* mOT *protocol enjoys* perfect forward secrecy (PFS) *(against passive and active attackers).*

Proof. The PFS case differs from the non-PFS case, proven in Theorem 1, in that, after completing the test session between Alice and Bob, the attacker is given the values $S_A = A^d$ and $S_B = B^d$, i.e., the private keys of the peers to the session. Only after receiving these values, the adversary needs to distinguish the test key from random.

Recall that due to the fact that the session key is the hash of the resulting secret value $\bar{K} = g^{2xye}$, where the hash is modeled as a random oracle, distinguishing the key is equivalent to finding $\bar{K}$. Thus, in the sequel we assume that a successful attacker is one that guesses this value g^{2xye}.

Examining the proof of Theorem 1 we can see that in all the simulations in that proof, the simulator knows the secret key S_A of Alice, and therefore it can provide it to the adversary upon corruption of Alice.

The difficulty in proving full PFS is the need to provide the attacker $\mathcal{M}$ with $S_B = B^d$ before $\mathcal{M}$ outputs its guess for the session key. This is very different than the case of Theorem 1 where the simulator first receives the attacker's guess and only then it uses this value to compute the forgery B^d. Still, with some significant changes to the simulation and some added assumptions, we will be able to prove the theorem by transforming a successful mOT attacker $\mathcal{M}$ into an RSA forger F that inverts RSA on a random input. We show this reduction now.

The RSA Forger F. The forger F is given as input an instance $N, e, R \in_R QR_N$ of the RSA problem and needs to compute R^d mod N with non-negligible probability. F starts by running a simulation of the mOT protocol against attacker $\mathcal{M}$ (which we assume to guess the test session key with non-negligible probability). For this F sets up the public parameters of mOT as N, e, g where N, e are from F's input and $g = h^e$ mod N for h chosen by F at random in QR_N. As in the proof of Theorem 1, F generates private keys for all parties except Bob[7] by programming the random oracle H (i.e., for each party id_P, F chooses $p \in_R QR_N$ and sets $H(id_P) = P = p^e$, and $S_P = p$).

For Bob, F chooses the value $B = H(id_B)$ by programming H as follows. It sets a value $U = g^u$ mod N where F chooses u as a random integer in the range $[1..\lfloor N/2 \rfloor]$ (note that with this choice of u, the distribution of the value U is statistically close to uniform in QR_N – by a similar argument as in Section 2).

[7] Bob is the peer to the test session - as in the previous proof we assume that the simulator successfully guesses the test session and its peers, an event that happens with non-negligible probability.

Then, it flips an unbiased coin *coin*, and sets

$$H(id_B) = B = \begin{cases} R^2 \bmod N & \text{if } \ coin = 0 \ \ \text{(where } R \text{ is part of } F\text{'s input)} \\ U^e \bmod N & \text{if } \ coin = 1 \end{cases}$$

Notice that the distribution of B is correct, i.e., (statistically close to) random in QR_N and independent of other values in the protocol. To simulate all the sessions other than the test session, F follows the same simulation as in Theorem 1. In other words, we keep the proof of Theorem 1 intact up to the point in that proof titled "Simulating the test session".

It remains to show how F simulates the test session interaction with $\mathcal{M}$ in the PFS case, and how this simulation results in the computation of R^d. For this we are going to first modify the attacker $\mathcal{M}$ into an attacker $\bar{\mathcal{M}}$ that behaves like $\mathcal{M}$ but, in addition, in runs where $\mathcal{M}$ guesses correctly the test session, $\bar{\mathcal{M}}$ will output some additional values that will allow F to complete its forgery. Thus, F will be running against the modified $\bar{\mathcal{M}}$ rather than against $\mathcal{M}$. We now show how we transform $\mathcal{M}$ into $\bar{\mathcal{M}}$ via several intermediate "games". We start by presenting a first game that represents the interaction with a KA-attacker in the test session experiment in the PFS setting.

The PFS Game. The following game represents the test session interaction between a "PFS challenger", $Chall_{PFS}$, and the KA-attacker $\mathcal{M}$ where $\mathcal{M}$ is allowed to corrupt Bob after the test session key is complete. In this case, $\mathcal{M}$ sends the incoming message β (allegedly coming from Bob) into the test session held by Alice, and Alice outputs a value α. After these values are set, the attacker receives Bob's secret key $S_B = B^d$ and $\mathcal{M}$ wins the game if it outputs $\bar{K} = g^{2exy}$.

PFS Game:

1. $Chall_{PFS}$ sends to $\mathcal{M}$ the values N, e, g, B, X.
   Here X represents the value α sent by Alice; indeed, since we assume that $\mathcal{M}$ can also be given S_A (which F chooses and hence knows) then providing $\alpha = X \cdot S_A$ is equivalent to providing X.
2. The adversary sends a value β which in turn determines a value Y defined as $Y = \beta / B^d$.
   Note that β, which is chosen by $\mathcal{M}$, may not be an element of QR_N, and the same holds for Y. However, $Y^2 \bmod N$ is necessarily in QR_N and hence generated by g. Defining $y = \log_g(Y^2 \bmod N)$, and thanks to the squaring operation in the computation of the session key in mOT, we get that the session key g^{2xye} equals X^{ye}.
3. $Chall_{PFS}$ sends B^d.
4. $\mathcal{M}$ sends $\bar{K}$. The adversary, $\mathcal{M}$, wins if $\bar{K} = X^{ye}$.

We now show how to transform an attacker $\mathcal{M}$ that wins the above PFS game (with non-negligible probability) into a modified attacker $\bar{\mathcal{M}}$ (with the same probability of success) that F will use to compute its RSA forgery. We arrive to $\bar{\mathcal{M}}$ from $\mathcal{M}$ through a series of adversaries $(\mathcal{M}, \mathcal{M}^*, \mathcal{M}_1, \mathcal{M}_1^*, \bar{\mathcal{M}})$ defined in the following games.

The adversary $\mathcal{M}^*$: Note that $\mathcal{M}$, in the above PFS game, can be changed to also output Y^{2e} (computed as β^{2e}/B^2). So in a winning run on input g, X, attacker $\mathcal{M}$ would output $\bar{K} = X^{ye}$ and also $Y^{2e} = g^{ye}$. By invoking the **KEA-DH** assumption there is another attacker $\mathcal{M}^*$ that behaves as $\mathcal{M}$ but in addition it outputs, in winning runs, $ye = \log_g Y^{2e}$ in step 4.

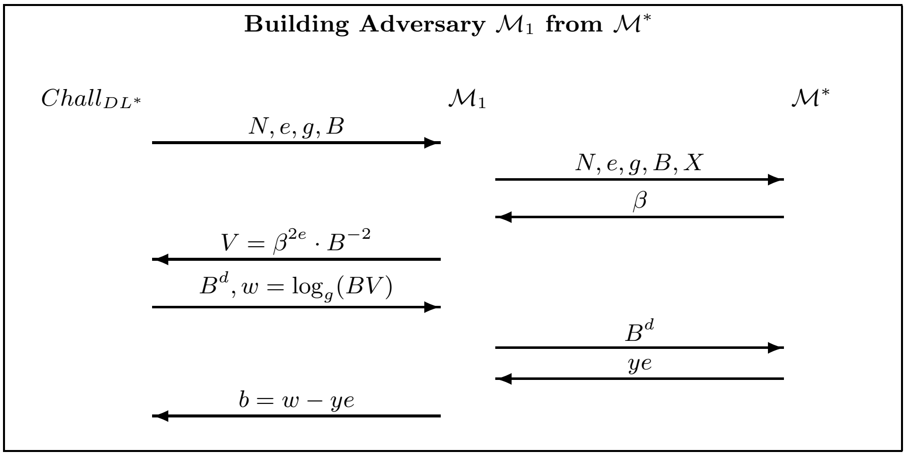

Fig. 2. Creating a **KEA-DL*** adversary

The adversary $\mathcal{M}_1$: We now use the modified PFS attacker $\mathcal{M}^*$ to build an attacker $\mathcal{M}_1$ that interacts with a challenger $Chall_{DL^*}$ in a **KEA-DL*** game. The actions of $\mathcal{M}_1$ are described in Figure 2: $\mathcal{M}_1$ uses $\mathcal{M}^*$ as a subroutine (where $\mathcal{M}_1$ acts as the PFS challenger with respect to $\mathcal{M}^*$) and uses responses received from $\mathcal{M}^*$ to answer $Chall_{DL^*}$ queries. Now, since in a successful run of $\mathcal{M}^*$ the last value ye output by $\mathcal{M}^*$ satisfies $g^{ye} = Y^{2e} = \beta^{2e}/B^2 = V$, and the value w output by $Chall_{DL^*}$ satisfies $w = \log_g BV = b + ye$, then the value $b = w - ye$ answered by $\mathcal{M}_1$ is correct (i.e., $g^b = B$) and $\mathcal{M}_1$ wins the KEA-DL* game. In other words, each successful run of $\mathcal{M}^*$ (which happens with non-negligible probability) induces a successful run of $\mathcal{M}_1$ in the **KEA-DL*** game.

The adversary $\mathcal{M}_1^*$: By the **KEA-DL*** Assumption, there is an adversary $\mathcal{M}_1^*$ that behaves exactly as $\mathcal{M}_1$ except that together with V it also outputs i, j such that $V = B^i g^j$. Replacing $\mathcal{M}_1$ with $\mathcal{M}_1^*$ in Figure 2, we get the same flows except that now the values i, j as above are added to the second flow from $\mathcal{M}_1$ to $Chall_{DL*}$. We refer to this as the *modified game of Figure 2.*

The hybrid adversary $\bar{\mathcal{M}}$: Using $\mathcal{M}_1^*$ we build an attacker $\bar{\mathcal{M}}$ that interacts in a PFS game as represented in Figure 3. In the first flow $\bar{\mathcal{M}}$ receives inputs from $Chall_{PFS}$ as in a regular PFS game. Next $\bar{\mathcal{M}}$ uses these inputs to call $\mathcal{M}_1^*$ in the modified game of Figure 2. In this modified game, $\mathcal{M}_1^*$ uses these same values as its first flow to $\mathcal{M}^*$. Upon receiving the β response from $\mathcal{M}^*$,

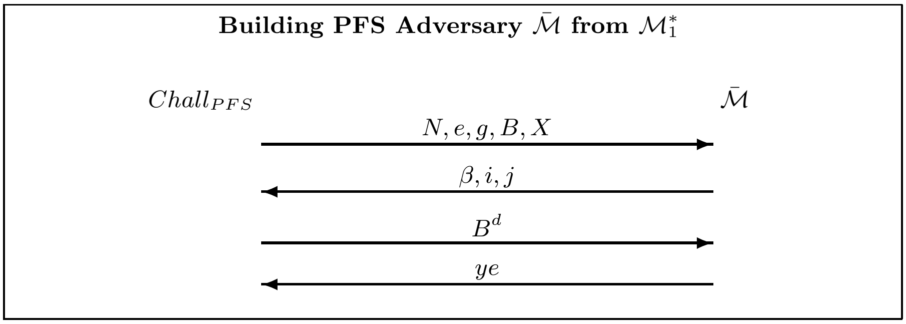
Building PFS Adversary $\bar{\mathcal{M}}$ from $\mathcal{M}_1^*$
$Chall_{PFS}$
$\bar{\mathcal{M}}$
N, e, g, B, X
β, i, j
B^d
ye

Fig. 3. A Hybrid PFS Adversary $\bar{\mathcal{M}}$ (values in second flow satisfy $\beta^{2e}B^{-2} = B^i g^j$)

$\mathcal{M}_1^*$ produces the value $V = \beta^{2e}B^{-2}$ as well as i, j such that $V = B^i g^j$. Then, $\bar{\mathcal{M}}$ sends to $Chall_{PFS}$ the values β, i, j (note that i, j are not part of the basic PFS game but additional outputs produced by $\bar{\mathcal{M}}$). In the next flow, $\bar{\mathcal{M}}$ receives from $Chall_{PFS}$ the value B^d as in a PFS game, which $\bar{\mathcal{M}}$ uses as the third flow from $\mathcal{M}_1^*$ to $\mathcal{M}^*$. Finally, upon receiving the response ye from $\mathcal{M}^*$, $\bar{\mathcal{M}}$ outputs this same value as its fourth message in the PFS game.

It is important to observe that the flows involving the value w produced by the dlog oracle in Figure 2 are *not* used by $\bar{\mathcal{M}}$; this is important since in a real execution of the PFS game – as part of the interaction with a mOT attacker – there is no such dlog oracle (this only exists as an artifact of the proof).

We note that in a successful run of $\bar{\mathcal{M}}$, the values β, i, j, ye output by $\bar{\mathcal{M}}$ satisfy the following equations:

$$g^{ye} = Y^{2e} = (\beta^e/B)^2 \bmod N \tag{3}$$

$$\beta^{2e}B^{-2} = V = B^i g^j \bmod N \tag{4}$$

which yield

$$g^{ye} = B^i g^j \bmod N \tag{5}$$

Also note that by virtue of the above sequence of games, if $\mathcal{M}$ succeeds with non-negligible probability then $\bar{\mathcal{M}}$ succeeds in outputting the above values also with non-negligible probability.

Summarizing the above transformations, we showed that the existence of a successful PFS attacker $\mathcal{M}$ implies (via the KEA assumptions) the existence of a second PFS attacker $\bar{\mathcal{M}}$ that in addition to the normal outputs of $\mathcal{M}$ (i.e., β and the session key guess), it also outputs the values i, j, ye that satisfy the above equations.

Applying the above to the key agreement setting of mOT we get that one can take a KA-attacker $\mathcal{M}$ that successfully breaks the PFS property of mOT and transform it into an equivalent KA-attacker $\bar{\mathcal{M}}$, that differs from $\mathcal{M}$ in that it also outputs the above values i, j, ye during a successful interaction in the test session experiment. We use this modified KA-attacker $\bar{\mathcal{M}}$ to complete the description of the forger F.

Back to the RSA Forger F. We have seen before how F simulates a run of mOT against a KA-attacker $\mathcal{M}$ except for the test session interaction. Here we complete the description of this part of the simulation and show how F computes its RSA forgery. For this we consider a run of F (as described so far) against the modified KA-adversary $\bar{\mathcal{M}}$ defined above; that is, $\bar{\mathcal{M}}$ behaves exactly as $\mathcal{M}$ but, in successful runs of $\mathcal{M}$, in addition to β it outputs i, j and in addition to the guess of the session key it outputs ye.

The way F uses $\bar{\mathcal{M}}$ to generate a forgery depends on two values: whether i output by $\bar{\mathcal{M}}$ is 0 or not, and whether the *coin* chosen by F (see above) is 0 or 1. We now show these different cases.

Case $i = 0$. With probability 1/2 we also have that $coin = 0$ and therefore $B = R^2 \bmod N$. In this case, F is running $\bar{\mathcal{M}}$ on $g = h^e$, $B = R^2$ and $X \in_R QR_N$. Assuming the run of $\bar{\mathcal{M}}$ is successful and $i = 0$, the forger F obtains the value $j = ye \bmod p'q'$ in the second message from $\bar{\mathcal{M}}$ in Figure 3 (recall that by (5) we have $ye = ib + j \bmod p'q'$). F then uses j to compute Y^2 as follows:

$$h^j = h^{ye} = (g^d)^{ye} = g^y = Y^2 \bmod N,$$

and uses Y^2 to compute $(R^4)^d = B^{2d} = (\beta^2/Y^2) \bmod N$. Using Lemma 1 and the fact that 4 and e are relatively prime F can compute, out of the value $(R^4)^d$, the required forgery $R^d \bmod N$.

Important: In the above case, F only needs to know j to complete its forgery; therefore it does not need to complete the third and fourth flows in Figure 3 (that involve B^d) before it can forge.

Case $i \neq 0$. With probability 1/2 we also have that $coin = 1$ and therefore $B = U^e = g^{ue} \bmod N$ where, by definition, u is a random integer in the range $[1..\lfloor N/2 \rfloor]$. In this case, F knows $B^d = U$ and hence can complete the full run against $\bar{\mathcal{M}}$ in Figure 3. That is, F runs $\bar{\mathcal{M}}$ sending (g, B, X) in the first message and U in the third message. F receives from $\bar{\mathcal{M}}$ the values i, j in the second message and $y' = ye$ in the last message. F will use i, j, y' to find two values s and t from which it will derive (whp) the factorization of N.

Specifically, F sets $s = ie$ and $t = y' - j$ (these operations are over the integers). It holds (mod $p'q'$) that

$$t = y' - j = ye - j = ib = ieu = su \bmod p'q'$$

where the third equality is from Equation (5) and $b = eu \bmod p'q'$ holds by the choice $B = U^e = g^{ue}$.

Thus, the integer $t - su$ is a multiple of $p'q'$ that F can compute as it knows s, t, u. If this number is non-zero then F factors N (as it is well known, the factorization of N can be found from such a multiple of $p'q'$, e.g. [20]). Now, note that there are at least two values of the integer u that will result in the same element U in QR_N (since $u \in_R [1..\lfloor N/2 \rfloor]$ and $\lfloor N/2 \rfloor \geq 2p'q'$). Since $\bar{\mathcal{M}}$ knows U but not the specific u (indeed, from the value U one cannot learn which of the possible values of u was chosen by F), the probability that t and s (that are derived from $\bar{\mathcal{M}}$ outputs) result in $t - su$ being zero is at most 1/2 (at most one of the u's can satisfy this equation).

In all, we have that if F interacts with a successful run of $\mathcal{M}$ (which implies a successful run of $\bar{\mathcal{M}}$) and $i = 0$ then with probability 1/2 F correctly computes R^d. If the run of $\mathcal{M}$ is successful and $i > 0$ then with probability 1/2 F factors N and hence can produce R^d as well. Since $\mathcal{M}$ (and $\bar{\mathcal{M}}$) succeed with non-negligible probability so does F in computing the forgery $R^d \bmod N$. $\square$

Acknowledgment. Research was sponsored by US Army Research laboratory and the UK Ministry of Defense and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

1. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004)
2. Blom, R.: An optimal class of symmetric key generation systems. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) EUROCRYPT 1984. LNCS, vol. 209, pp. 335–338. Springer, Heidelberg (1985)
3. Boyd, C., Choo, K.-K.R.: Security of two-party identity-based key agreement. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 229–243. Springer, Heidelberg (2005)
4. Boyd, C., Mao, W., Paterson, K.G.: Key Agreement Using Statically Keyed Authenticators. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 248–262. Springer, Heidelberg (2004)
5. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
6. Chen, L., Cheng, Z., Smart, N.P.: Identity-based key agreement protocols from pairings. Int. J. Inf. Sec. 6(4), 213–241 (2007)
7. Chen, L., Kudla, C.: Identity Based Authenticated Key Agreement Protocols from Pairings. In: 16th IEEE Computer Security Foundations Workshop - CSFW 2003, pp. 219–233. IEEE Computer Society Press, Los Alamitos (2003)
8. Damgård, I.: Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)
9. De Santis, A., Desmedt, Y., Frankel, Y., Yung, M.: How to share a function securely. In: STOC '94, pp. 522–533. ACM Press, New York (1994)
10. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Trans. Info. Theor. 22(6), 644–654 (1976)
11. Goldreich, O., Rosen, V.: On the security of modular exponentiation with application to the construction of pseudorandom generators. Journal of Cryptology 16(2), 71–93 (2003)

12. Gunther, C.G.: An Identity-Based Key-Exchange Protocol. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 29–37. Springer, Heidelberg (1990)
13. Hada, S., Tanaka, T.: On the Existence of 3-round Zero-Knowledge Protocols. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, p. 408. Springer, Heidelberg (1998)
14. Hastad, J., Schrift, A., Shamir, A.: The Discrete Logarithm Modulo a Composite Hides O(n) Bits. J. Comput. Syst. Sci. 47(3), 376–404 (1993)
15. Jarecki, S., Kim, J., Tsudik, G.: Beyond Secret Handshakes: Affiliation-Hiding Authenticated Key Exchange. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 352–369. Springer, Heidelberg (2008)
16. Krawczyk, H.: SKEME: A Versatile Secure Key Exchange Mechanism for Internet. In: 1996 Internet Society Symposium on Network and Distributed System Security, NDSS (1996)
17. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
18. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An efficient Protocol for Authenticated Key Agreement. Designs, Codes and Cryptography 28, 119–134 (2003)
19. McCullagh, N., Barreto, P.S.L.M.: A New Two-Party Identity-Based Authenticated Key Agreement. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 262–274. Springer, Heidelberg (2005)
20. Menezes, A., Van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
21. Okamoto, E.: Key Distribution Systems Based on Identification Information. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 194–202. Springer, Heidelberg (1988)
22. Okamoto, E., Tanaka, K.: Key Distribution System Based on Identification Information. IEEE Journal on Selected Areas in Communications 7(4), 481–485 (1989)
23. Shamir, A.: On the Generation of Cryptographically Strong Pseudorandom Sequences. ACM Trans. Comput. Syst. 1(1), 38–44 (1983)
24. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
25. Schridde, C., Smith, M., Freisleben, B.: An Identity-Based Key Agreement Protocol for the Network Layer. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 409–422. Springer, Heidelberg (2008)
26. Shmuely, Z.: Composite Diffie-Hellman Public-Key Generating Systems are Hard to Break, Technical Report 356, CS Dept., Technion, Israel (1985)
27. Shoup, V.: On formal models for secure key exchange (version 4) (November 15, 1999), `http://www.shoup.net/papers/`
28. Smetters, D.K., Durfee, G.: Domain-based Administration of Identity-Based Cryptosystems for Secure E-Mail and IPSEC. In: SSYM 2003: Proceedings of the 12th Conference on USENIX Security Symposium, Berkeley, CA, USA, p. 15. USENIX Association (2003)
29. Wang, Y.: Efficient Identity-Based and Authenticated Key Agreement Protocol. Cryptology ePrint Archive, Report 2005/108 (2005), `http://eprint.iacr.org/2005/108/`