Reusing Static Keys in Key Agreement Protocols

Sanjit Chatterjee
(Joint work with Alfred Menezes and Berkant Ustaoglu)

University of Waterloo

May 14, 2009

To Reuse or Not To Reuse

Robustness principles for public key protocols:

> *If possible avoid using the same key for two different purposes (such as signing and decryption) ...*

Handbook of Applied Cryptography:

> *The principle of* key separation *is that keys for different purposes should be cryptographically separated.*

But reuse can be beneficial!

- Cost effective
- Efficient

Reuse: Pitfalls

- Kelsey, Schneier and Wagner [1998]
  - *Chosen protocol attack:* Design a new protocol to attack an existing protocol when the keying material is shared.
- Gligoroski, Andova and Knapskog [2008]
  - Using the same key for CBC and OFB/CTR mode of operation can be detrimental.

Reuse: Not Necessarily Bad

- Coron et al. [2002]
  - RSA key pairs can be reused for PSS versions of signature and encryption.
- Vasco et al. [2008]
  - Boneh-Franklin IBE and Hess's Id-based signature.
  - Pointcheval-Stern version of ElGamal signature and ElGamal encryption with Fujisaki-Okamoto conversion.

What's NIST saying

A static key pair may be used in more than one key establishment scheme. However, one static public/private key pair **shall not** *be used for different purposes (for example, a digital signature key pair is not to be used for key establishment or vice versa) with the following possible exception: when requesting the (initial) certificate for a public static key establishment key, the key establishment private key associated with the public key may be used to sign certificate request.*

NIST SP 800-56A, March, 2007

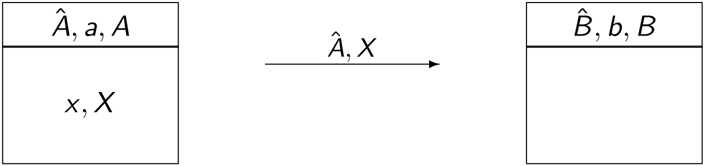
The Setting

- Variants of Diffie-Hellman protocol.
- $G = \langle g \rangle$: cyclic group of prime order q.
- $\hat{A}$: *Initiator* ($\mathcal{I}$)
  - Static key pair: $(a \in_R \mathbb{Z}_q^*, A = g^a)$
  - Ephemeral key pair: $(x \in_R \mathbb{Z}_q^*, X = g^x)$
- $\hat{B}$: *Responder* ($\mathcal{R}$)
  - Static key pair: $(b,\ B)$
  - Ephemeral key pair: (y, Y)
- CA: Issues certificates binding a party's identifier to its static public key.

Unified Model

- Family of two-party Diffie-Hellman key agreement protocols.
- Standardized in ANSI X9.42, ANSI X9.63, NIST SP 800-56A.

One-pass Unified Model



$$\kappa_1 = H(g^{xb}, g^{ab}, \mathrm{keydatalen}, \mathrm{AlgorithmID}, \hat{A}, \hat{B}, \Lambda)$$

- **keydatalen:** Bitlength of secret keying material to be generated.
- **AlgorithmID:** How the derived keying material will be parsed and for which algorithm(s) it will be used
- Λ: Optional public information.

Three-pass Unified Model

Two pass protocol combined with bilateral key confirmation.

$\hat{A}, a, A$		$\hat{B}, b, B$
x, X	$\xrightarrow{\hat{A}, X}$	y, Y
	$\xleftarrow{\hat{B}, Y, T_B = \mathrm{MAC}_{\kappa'}(\mathcal{R}, \hat{B}, \hat{A}, Y, X, \Lambda_1)}$	
	$\xrightarrow{T_A = \mathrm{MAC}_{\kappa'}(\mathcal{I}, \hat{A}, \hat{B}, X, Y, \Lambda_2)}$	

$$(\kappa', \ \kappa_2) = H(g^{xy}, g^{ab}, \text{keydatalen}, \text{AlgorithmID}, \hat{A}, \hat{B}, \Lambda)$$

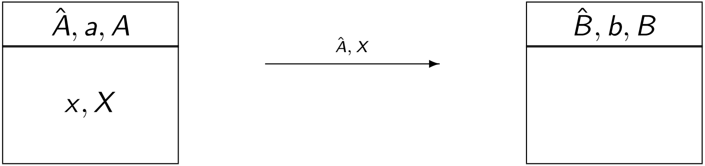
- $\Lambda_1, \ \Lambda_2$: Optional public strings
- κ_2 : Session key
- κ' : Ephemeral secret key

The Backdrop

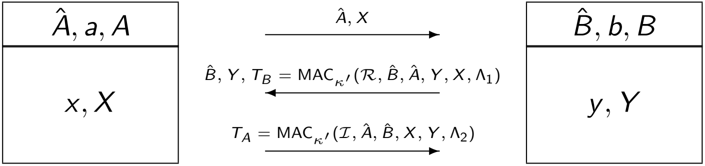
- One-pass protocol is used to derive 256 bits session key $\kappa_1 = (\kappa_m,\ \kappa_e)$.
  - κ_m : 128 bit HMAC key
  - κ_e : 128-bit AES key
- Three-pass protocol uses HMAC with 128-bit κ' and produces 128-bit session key κ_2.
- Attacker is able to use a SessionKeyReveal to obtain session keys produced by the one-pass protocol.
- Both protocols use same AlgorithmID.

NIST SP 800-56A:

> AlgorithmID *might indicate that bits* $1 - 80$ *are to be used as an* 80-*bit HMAC key and that bits* $81 - 208$ *are to be used as a* 128-*bit AES key.*



$$\kappa_1 = H(g^{xb}, g^{ab}, \text{keydatalen}, \text{AlgorithmID}, \hat{A}, \hat{B}, \Lambda)$$



$$(\kappa', \ \kappa_2) = H(g^{xy}, g^{ab}, \text{keydatalen}, \text{AlgorithmID}, \hat{A}, \hat{B}, \Lambda)$$

The Attack

1. $\mathcal{M}$ initiates a session sid_1 of three-pass UM at $\hat{A}$; receives $(\hat{A}, X)$.
2. $\mathcal{M}$ forwards $(\hat{A}, X)$ to $\hat{B}$ in a session sid_2 of one-pass UM.
3. $\hat{B}$ computes session key κ_2 following one-pass UM.
4. $\mathcal{M}$ issues a *SessionKeyReveal* to sid_2 at $\hat{B}$ to obtain $\kappa_1 = (\kappa_m,\ \kappa_e)$.
5. $\mathcal{M}$ sets $Y = B$, so $\kappa_1 = (\kappa',\ \kappa_2)$ under our assumptions. $\mathcal{M}$ computes T_B, sends $(\hat{B}, Y, T_B)$ to session sid_1 at $\hat{A}$.
6. $\hat{A}$ computes κ_2 in sid_1 which is known to $\mathcal{M}$.

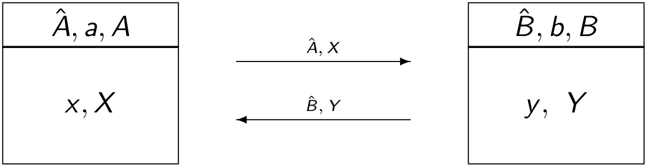
Same attack can be launched against three-pass MQV when static keys are reused with one-pass MQV.

KEA+h Protocol

KEA: Autheticated key exchange protocol; introduced by NSA.

KEA+: Modification of KEA; introduced by Lauter-Mityagin [PKC2006].

KEA+h: Modification of KEA+.



$$\kappa = H(H_1(g^{ay}), H_1(g^{bx}), \hat{A}, \hat{B})$$

τ-Protocol

A new protocol.

Uses an MTI/C0-like exchange of messages to confirm the receipt of ephemeral public keys.

Can be proven secure in the Canetti-Krawczyk model.

$$
\begin{array}{|c|}\hline \hat{A}, a, A \\ \hline x, X \\ \hline \end{array}
\quad
\begin{array}{c}
\xrightarrow{\hat{A},\, X,\, T_A = H_2(g^{ab}, X, \hat{A}, \hat{B}, \mathcal{I})} \\
\xleftarrow{\hat{B},\, Y,\, T_B = H_2(g^{ab}, Y, \hat{B}, \hat{A}, \mathcal{R}),\, \overline{X} = H_1(X^b)} \\
\xrightarrow{\overline{Y} = H_1(Y^a)}
\end{array}
\quad
\begin{array}{|c|}\hline \hat{B}, b, B \\ \hline y, Y \\ \hline \end{array}
$$

$$\kappa = H(g^{xy},\ X,\ Y)$$

The Scenario

- Attack a KEA+h session using τ-protocol.
- $\hat{A}$ uses the KEA+h protocol in a stand-alone setting.
- $\hat{B}$ uses the same static key for KEA+h and τ.
- $\hat{A}$ initiates a KEA+h session with $\hat{B}$.
- $\hat{A}$ ends up getting her session key compromised.

$\hat{A}, a, A$		$\hat{B}, b, B$
x, X	$\xrightarrow{\hat{A}, X}$ $\xleftarrow{\hat{B}, Y}$	y, Y

$$\kappa = H(H_1(g^{ay}), H_1(g^{bx}), \hat{A}, \hat{B})$$

$\hat{A}, a, A$		$\hat{B}, b, B$
x, X	$\xrightarrow{\hat{A}, X, T_A = H_2(g^{ab}, X, \hat{A}, \hat{B}, \mathcal{I})}$ $\xleftarrow{\hat{B}, Y, T_B = H_2(g^{ab}, Y, \hat{B}, \hat{A}, \mathcal{R}), \overline{X} = H_1(X^b)}$ $\xrightarrow{\overline{Y} = H_1(Y^a)}$	y, Y

$$\kappa = H(g^{xy}, X, Y)$$

The Attack

1. $\mathcal{M}$ initiates a KEA+h session at $\hat{A}$ with $\hat{B}$ as the peer and obtains the outgoing ephemeral public key X.
2. $\mathcal{M}$ controls a party $\hat{E}$ with static key pair $(e,\ E = g^e)$ and initiates a τ session with $\hat{B}$ by sending the message $X, T_E = \left(H_2(B^e, X, \hat{E}, \hat{B}, \mathcal{I})\right)$
3. $\hat{B}$ responds with $(Y, T_B, H_1(X^b))$ from which $\mathcal{M}$ obtains $H_1(X^b)$.
4. $\mathcal{M}$ selects an ephemeral key pair $(z, Z = g^z)$ and sends $(\hat{B},\ Z)$ to $\hat{A}$ in KEA+h.
5. $\hat{A}$ computes the KEA+h session key as $\kappa = H\left(H_1(Z^a), H_1(B^x), \hat{A}, \hat{B}\right)$.
6. $\mathcal{M}$ computes the same session key as $\kappa = H\left(H_1(A^z), H_1(X^b), \hat{A}, \hat{B}\right)$.

"Shared" Model

- To capture the security assurances guaranteed by two (or more) distinct key agreement protocols.
- Each party uses same static key pair in all the protocols.
- Individual protocols are two-party Diffie-Hellman variety.
- Enhances the extended Canetti-Krawczyk model.
  - $\Pi_1,\ \Pi_2, \ldots, \Pi_d$ are run concurrently by a party.
  - Same static key is reused for all the protocols.
  - Π_i provides the security attributes implied by the eCK model.

Security Model

Essential idea: add a protocol identifier

- Protocol message: $(\Pi_i,\ \hat{A},\ \hat{B},\ \mathit{role},\ \mathit{Comm})$.
- Session identifier: $(\Pi_i,\ \hat{A},\ \hat{B},\ \mathit{role},\ \ldots)$.

Matching session

- $\mathit{sid} = (\Pi_i, \hat{A},\ \hat{B},\ \mathit{role}_A, \mathit{Comm}_A)$
- $\mathit{sid}^* = (\Pi_j, \hat{C},\ \hat{D},\ \mathit{role}_C, \mathit{Comm}_C)$
- *matching* if $\Pi_i = \Pi_j,\ \hat{A} = \hat{D},\ \hat{B} = \hat{C},\ \mathit{role}_A \neq \mathit{role}_C$ and $\mathit{Comm}_A \equiv \mathit{Comm}_C$

Adversary

Modeled as a probabilistic Turing machine $\mathcal{M}$ and controls *all* communications.

$\mathcal{M}$ can make the following queries:

- StaticKeyReveal($\hat{A}$)
- *EphemeralKeyReveal*(*sid*)
- *SessionKeyReveal*(*sid*)
- *EstablishParty*($\hat{A}$, A)
    - To model attacks by malicious insiders.
    - Parties established by $\mathcal{M}$ are *corrupted*.
    - A party not corrupted is *honest*.

Adversary's Goal

- $\mathcal{M}$ is allowed to make a special query *Test*(*sid*) to a 'fresh' session *sid*.
- $\mathcal{M}$ is given with equal probability either the session key of *sid* or a random key.
- $\mathcal{M}$ wins if its guess is correct.
- $\mathcal{M}$ can continue interacting with the parties after issuing the *Test* query, but the test session must remain fresh.

Π-fresh

- *sid*: A completed Π-session; owner $\hat{A}$, peer $\hat{B}$: both honest.
- sid^*: Matching session of *sid*, if exists.
- *sid* is Π-fresh if *none* of the following conditions hold:
  1. $\mathcal{M}$ issued *SessionKeyReveal*(*sid*) or *SessionKeyReveal*(sid^*).
  2. (sid^*) exists and $\mathcal{M}$ issued one of the following:
     2.1 Both *StaticKeyReveal*($\hat{A}$) and *EphemeralKeyReveal*(*sid*).
     2.2 Both *StaticKeyReveal*($\hat{B}$) and *EphemeralKeyReveal*(sid^*).
  3. (sid^*) does not exist and $\mathcal{M}$ issued one of the following:
     3.1 Both *StaticKeyReveal*($\hat{A}$) and *EphemeralKeyReveal*(*sid*).
     3.2 *StaticKeyReveal*($\hat{B}$).

Security in Shared Model

$\Pi_1, \Pi_2, \ldots, \Pi_d$: *secure in the shared model if the following conditions hold:*

1. *If two honest parties complete matching Π_i-sessions then, except with negligible probability, they both compute the same session key.*
2. *No $\mathcal{M}$ can distinguish the session key of a fresh Π_i-session from a randomly chosen session key, with probability greater than $\frac{1}{2}$ plus a negligible fraction.*

NAXOS-C

$\hat{A}, a, A$		$\hat{B}, b, B$
$\tilde{x}, X = g^{H_1(a,\ \tilde{x})}$	$\xrightarrow{\hat{A}, X}$	$\tilde{y}, Y = g^{H_1(b, \tilde{y})}$
	$\xleftarrow{\hat{B},\ Y,\ T_B = H_2(\kappa_m,\ \mathcal{R}, \hat{B}, \hat{A}, Y, X, \Pi_1)}$	
	$\xrightarrow{\hat{A},\ T_A = H_2(\kappa_m,\ \mathcal{I}, \hat{A}, \hat{B}, X, Y, \Pi_1)}$	

$$(\kappa_m,\ \kappa) = H(g^{ay},\ g^{bx},\ \hat{A}, \hat{B}, X, Y, \Pi_1)$$

- $H : \{0,\ 1\}^* \to \{0,\ 1\}^\gamma \times \{0,\ 1\}^\gamma$
- $H_1 : \{0,\ 1\}^* \to [0,\ q-1]$
- $H_2 : \{0,\ 1\}^* \to \{0,\ 1\}^{2\gamma}$

DHKEA

$\hat{A}, a, A$		$\hat{B}, b, B$
$\tilde{x}, X = g^{H_1(a,\ \tilde{x})}$	$\xrightarrow{\hat{A},\, X}$	$\tilde{y}, Y = g^{H_1(b, \tilde{y})}$
	$\xleftarrow{\hat{B},\, Y,\, T_B \,=\, H_2(\kappa_m,\ g^{bx},\ \mathcal{R}, \hat{B}, \hat{A}, Y, X, \Pi_2)}$	
	$\xrightarrow{\hat{A},\ T_A \,=\, H_2(\kappa_m,\ g^{ay},\ (\mathcal{I}, \hat{A}, \hat{B}, X, Y, \Pi_2)}$	

$$(\kappa_m,\ \kappa) = H(g^{xy},\ \hat{A}, \hat{B}, X, Y, \Pi_2)$$

- $H : \{0,\ 1\}^* \rightarrow \{0,\ 1\}^\gamma \times \{0,\ 1\}^\gamma$
- $H_1 : \{0,\ 1\}^* \rightarrow [0,\ q-1]$
- $H_2 : \{0,\ 1\}^* \rightarrow \{0,\ 1\}^{2\gamma}$

Conclusion

- Vulnerabilty of KA protocols when static key is reused.
- Security model for KA protocols allowing such reuse.

Our shared model assumes each party has exactly one static key pair.

Further refinements:

1. Each party having mutiple static key pairs.
2. Protocols having different security attributes.

What's your conclusion?