Improved Deniable Signature Key Exchange for mpOTR

Document Revision: 21a0c24c1c8e+

Matthew Van Gundy <matt@singlethink.net>

April 8, 2013

The mpOTR session setup (esp deniable signature key exchange) is very inefficient in terms of the number of cryptographic operations that must be performed. The following table compares the number of operations that *each* participant must perform. For a group of n participants, each participant must perform:

Operation	Original mpOTR [2]	Deniable GKA + DSKE
Messages	$12 \cdot n$	$4n$
Hash	$n + 9$	7
Symmetric Key Generation	$n + 2$	1
Symmetric Encryption	$2 \cdot n + 1$	0
Asym. Encryption / Modular Exp.	$4 \cdot n + 2$	$2 \cdot n + 4$
Signature Key Generation	1	1
Signature	$n + 1$	1

Instead of doing a pairwise Deniable Signature Key Exchange as described in the paper, we can augment a Deniable Group Key Agreement like Bohli's [1] to also exchange an ephemeral signature key. Augmentations to achieve ephemeral key exchange are shown in boxes.

	Bohli's GKA [1] for instance Π_i of principal U_i $\boxed{\text{w/ DSKE}}$
Round 1: Generate key share	$k_i \xleftarrow{\$} \{0,1\}^k$
$\boxed{\text{Generate ephemeral signature key}}$	$\boxed{(S_i, s_i) \xleftarrow{\$} SIG.Gen()}$
Generate DH key for circular gka	$x_i \xleftarrow{\$} Z_q, y_i = g^{x_i}$
Commit to key share	$M_i^1 = (H(k_i), y_i, U_i, \boxed{S_i})$
Round 2: Compute session id	$\text{sid}_i = H(\text{pid}_i \|\| H(k_1) \|\| \ldots \|\| H(k_n))$
Generate Schnorr blind	$r_i \xleftarrow{\$} Z_q, z_i = g^{r_i}$
Broadcast	$M_i^2 = (\text{sid}_i, z_i, U_i)$
Round 3: Compute neighbor secrets	$t_i^L = H(y_{i-1}^{x_i}), t_i^R = H(y_{i+1}^{x_i}), T_i = t_i^L \oplus t_i^R$
Send Key Share	$M_i^3 = (k_i \oplus t_i^R, T_i, U_i)$
Round 4: Verify	$T_1 \oplus \cdots \oplus T_n = 0$ and for all decrypted k_j, $H(k_j)$ equals the 1st component of $M_j^1 (j \in \{1, \ldots, n\} \setminus \{i\})$
	$S_1 \neq S_j \neq \cdots \neq S_n$ for all $j \in \{2, \ldots, n-1\}$
Session key	$\text{sk}_i = H(\text{pid}_i \|\| k_1 \|\| \ldots \|\| k_n)$
Session Confirmation	$\text{sconf}_i = H((y_1, k_1, \boxed{S_1}) \|\| \ldots \|\| (y_n, k_n, \boxed{S_n}))$
Schnorr Challenge / Proof	$c_i = H(\text{sid}_i \|\| \text{sconf}_i) \bmod q, d_i = r_i - c_i \alpha_i \bmod q$
$\boxed{\text{Signature Key Confirmation}}$	$\boxed{\sigma_i = SIG.Sign(s_i, c_i)}$
Send Proof	$M_i^4 = (d_i, U_i, \boxed{\sigma_i})$
Verify	$g^{d_j} (PK_j)^{c_i} = z_j$ for all $j \in \{1, \ldots, n\} \setminus \{i\}$
$\boxed{\text{Signature Key Confirmation}}$	$\boxed{SIG.Verify(S_j, c_i, \sigma_j) \text{ for all } j \in \{1, \ldots, n\} \setminus \{i\}}$

In the original protocol, all participants prove their identity and mutually authenticate their key shares by means of a zero-knowledge proof of knowledge over the set of participants, their key share commitments, and their individual key share contributions. To authenticate ephemeral signature keys, we also cover the ephemeral signature keys under the ZKP. Each participant also proves knowledge of the private key associated with their ephemeral signature key by signing the random challenge (c_i) for this session. I have not yet extended the proof in [1], however using c_i as the message to sign should not affect the security of the Schnorr Zero-Knowledge Proof in any way — c_i is not secret, it just should not be predictable until after the prover has committed to her random value (k_i in the above protocol). We ensure that one user cannot impersonate another by ensuring that all ephemeral signature keys are unique.

References

[1] Jens-Matthias Bohli and Rainer Steinwandt. Deniable Group Key Agreement. In Phong Q. Nguyen, editor, *Progress in Cryptology – VIETCRYPT 2006*, volume 4341 of *LNCS*, pages 298–311. Springer-Verlag, 2006.

[2] Ian Goldberg, Berkant Ustaoglu, Matthew Van Gundy, and Hao Chen. Multi-party Off-the-Record Messaging. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, New York, NY, USA, November 2009. ACM Press.