Auxiliary Encryption in ElectionGuard as introduced in version 1.0 of the specification

Michael Naehrig

January 2022

Notation is as in the ElectionGuard specification v0.95. Let g be the generator of the ElectionGuard group, i.e. $G = \langle g \rangle$ is the multiplicative group of prime order q generated by g modulo the prime p. The number of guardians is $n \in \mathbb{N}$, $s_i \in \mathbb{Z}_q$ is the secret key of guardian i and $K_i = g^{s_i} \mod p$ is the corresponding public key. The election master public key is $K = K_1 \cdot K_2 \cdot \cdots \cdot K_n = g^{s_1 + \cdots + s_n} \mod p$.

1 Encrypting key shares and auxiliary data

According to the current specification v0.95, each guardian generates an auxiliary key pair for an auxiliary public key encryption scheme. This is used to receive key shares from the other guardians in encrypted form, but can also be used to encrypt other data. If this auxiliary encryption is a variant of ElGamal, the guardian's ElGamal key pair (s_ℓ, K_ℓ) (for guardian T_ℓ with number ℓ) can be used.

For the auxiliary public key encryption we use *hashed ElGamal*, which deploys a key derivation function (KDF) to generate a key stream that is then xored with the data. To implement the KDF and to provide a message authentication code (MAC), the encryption requires the keyed Hash Message Authentication Code HMAC (as defined in NIST PUB FIPS 198-1) instantiated with the hash function H = SHA-256, i.e. HMAC-SHA-256. HMAC takes as input a secret key k and a message (or data) m and returns a bit string $\mathrm{HMAC}(k, m)$ of the same length as the output of the hash function H, i.e. 256 bits (32 bytes) for SHA-256.

1.1 Encryption to a guardian's public key using hashed ElGamal

Assume that the length of the data m to be encrypted in bytes is a multiple of 32. If this is not the case, m is padded to a multiple of 32 by an appropriate padding method. Divide m into L 32-byte blocks m_i,

$$m = m_1 || m_2 || \ldots || m_L.$$

To encrypt m to the public key K_ℓ, generate an ElGamal pair $(a, b) = (g^R, K_\ell^R)$ as usual and compute a secret key

$$k = H(a, b) = H(g^R, K_\ell^R) \in \{0,1\}^{256}.$$

The key k is now used for a KDF in counter mode (as specified in NIST SP 800-108) to generate a MAC key k_0 and an encryption key $k_1 || \ldots || k_L$, where

$$k_i = \mathrm{HMAC}(k, [i]_{32} || \bar{Q} || [L]_{32}).$$

Here, $[i]$ is a 32-bit encoding of the counter i, $[L]$ is a 32-bit encoding of the length of the data to be encrypted, and $\bar{Q}$ is the extended base hash. The ciphertext is then computed as

$$\begin{aligned} c = E_\ell(R, m) &= (c_0, c_1, c_2), \\ c_0 &= g^R, \\ c_1 &= m_1 \oplus k_1 || m_2 \oplus k_2 || \dots || m_L \oplus k_L, \\ c_2 &= \text{HMAC}(k_0, c_0 || c_1). \end{aligned}$$

The component c_2 of the ciphertext is a MAC computed from the hashed ElGamal encryption (c_0, c_1) to detect unauthorized modification of the ciphertext. A ciphertext consists of one group element modulo p, a byte array as long as the encrypted data and one 256-bit MAC value. For the 4096-bit ElectionGuard prime p and an encryption of a 128-byte auxiliary text field, the ciphertext is $512 + 128 + 32 = 672$ bytes.

1.2 Decryption of hashed ElGamal

Decryption first uses either the secret key s_ℓ or the random nonce R to compute $K_\ell^R = (g^R)^{s_\ell}$. Next, $k = H(c_0, K_\ell^R)$ is computed and the MAC key k_0 is derived as above. It is then checked whether $c_2 = \text{HMAC}(k_0, c_0 || c_1)$. Decryption aborts if this is not the case. Then, the key blocks $k_i = \text{HMAC}(k, [i] || \bar{Q} || [L])$ are obtained for $i = 1, \dots, L$ as above and the message can be decrypted via $m = c_1 \oplus (k_1 || \dots || k_L)$.

1.3 Encrypting key shares with hashed ElGamal

Guardian T_i encrypts their key share $P_i(\ell)$ to guardian T_ℓ by encrypting to T_ℓ's public key K_ℓ. Since $P_i(\ell) \in \mathbb{Z}_q$, the encrypted data here is only one block long. The ciphertext in this case is

$$\begin{aligned} c = E_\ell(R, m) &= (c_0, c_1, c_2), \\ c_0 &= g^R, \\ c_1 &= [P_i(\ell)]_{256} \oplus k_1, \\ c_2 &= \text{HMAC}(k_0, c_0 || c_1), \end{aligned}$$

where $k_j = \text{HMAC}(k, [j]_{32} || \bar{Q} || [L]_{32})$ for $jS \in \{0, 1\}$ is as above and $[P_i(\ell)]_{256}$ is a 256-bit encoding of $P_i(\ell)$. Decryption works as above.

1.4 Encrypting to the election public key

Auxiliary data on a ballot such as information about over-votes or write-in candidates is encrypted to the election master public key K. It is done as described above using K instead of K_ℓ. Verifiable decryption with all n guardians or a quorum of k guardians is done in the usual way just as for decrypting other data. Guardian T_i computes

$$M_i = c_0^{s_i}$$

or it is reconstructed for them if they are not present. This allows to compute

$$K^R = \prod_{i=1}^{n} M_i.$$

From here, decryption proceeds by hashing $k = H(c_0, K^R)$ and so on as above.