CISC 468: CRYPTOGRAPHY

LESSON 14: THE ELGAMAL DIGITAL SIGNATURE SCHEME

Furkan Alaca

READINGS

- Section 10.3: The Elgamal Digital Signature Scheme, Paar & Pelzl
- Section 10.4: The Digital Signature Algorithm (DSA), Paar & Pelzl

INTRODUCTION

- The RSA encryption and digital signature operations are nearly the same
- Elgamal digital signatures are based on the difficulty of computing discrete logarithms, but the signature operation is quite different from Elgamal encryption
- The Digital Signature Algorithm (DSA), which is published by NIST and is the most widely used digital signature standard, is a variant of the Elgamal signature algorithm

SCHOOLBOOK ELGAMAL DIGITAL SIGNATURE: SETUP

The public-private key pair is computed in a setup phase, which is identical to Elgamal encryption:

1. Choose a large prime p.
2. Choose a primitive element α of $\mathbb{Z}_p^*$ or a subgroup of $\mathbb{Z}_p^*$.
3. Choose a random integer $d \in \{2, 3, \ldots, p-2\}$.
4. Compute $\beta = \alpha^d \bmod p$.

The public key is formed by $k_{pub} = (p, \alpha, \beta)$,
and the private key by $k_{pr} = d$.

SCHOOLBOOK ELGAMAL DIGITAL SIGNATURE: SIGNING

To compute the signature $\mathrm{sig}_{k_{pr}}(x, k_E)$ for a message x:

1. Choose a random ephemeral key $k_E = \{0, 1, 2, \ldots, p-2\}$ such that $\gcd(k_E, p-1) = 1$.
2. Compute the signature $\mathrm{sig}_{k_{pr}}(x, k_E) = (r, s)$, where:

$$r \equiv \alpha^{k_E} \mod p,$$

$$s \equiv (x - d \cdot r) k_E^{-1} \mod p - 1.$$

SCHOOLBOOK ELGAMAL DIGITAL SIGNATURE: VERIFICATION

The receiver runs $ver_{k_{pub}}(x,(r,s))$ to verify the signature (r,s):

1. Compute the value $t \equiv \beta^r \cdot r^s \bmod p$.
2. If $t \equiv \alpha^x \bmod p$, the signature is valid. Otherwise, it is invalid.

In other words, the verifier accepts the signature only if

$$\beta^r \cdot r^s \equiv \alpha^x \bmod p.$$

SCHOOLBOOK ELGAMAL DIGITAL SIGNATURE: PROOF

Step 1 of verification is to compute t:

$$\begin{aligned} t &\equiv \beta^r \cdot r^s \bmod p \\ &\equiv (\alpha^d)^r \cdot (\alpha^{k_E})^s \bmod p \\ &\equiv \alpha^{dr+k_E s} \bmod p \end{aligned}$$

SCHOOLBOOK ELGAMAL DIGITAL SIGNATURE: PROOF (2)

Step 2 is to check that $t \equiv \alpha^x \bmod p$:

$$t \equiv \alpha^{d \cdot r + k_E \cdot s} \stackrel{?}{\equiv} \alpha^x \bmod p$$

By Fermat's Little Theorem, the equality holds if the exponents on both sides are congruent $\bmod p - 1$:

$$x \stackrel{?}{\equiv} d \cdot r + k_E \cdot s \bmod p - 1$$

Rearranging the above expression gives us the formula for computing the signature, $s \equiv (x - d \cdot r)k_E^{-1} \bmod p - 1.$

SCHOOLBOOK ELGAMAL DIGITAL SIGNATURE: EXAMPLE

Bob signs and sends the message $x = 26$ to Alice:

Alice		**Bob**
		1. choose $p = 29$
		2. choose $\alpha = 2$
		3. choose $d = 12$
		4. $\beta = \alpha^d \equiv 7 \bmod 29$
	$\xleftarrow{(p,\alpha,\beta)=(29,2,7)}$	
		compute signature for message $x = 26$: choose $k_E = 5$, note that $\gcd(5, 28) = 1$ $r = \alpha^{k_E} \equiv 2^5 \equiv 3 \bmod 29$ $s = (x - d\,r)\,k_E^{-1} \equiv (-10) \cdot 17 \equiv 26 \bmod 28$
	$\xleftarrow{(x,(r,s))=(26,(3,26))}$	
verify: $t = \beta^r \cdot r^s \equiv 7^3 \cdot 3^{26} \equiv 22 \bmod 29$ $\alpha^x \equiv 2^{26} \equiv 22 \bmod 29$ $t \equiv \alpha^x \bmod 29 \Longrightarrow$ valid signature		

ELGAMAL DIGITAL SIGNATURES: COMPUTATIONAL ASPECTS

Computational aspects that we covered previously still apply:

- The security relies on the discrete logarithm problem, so p should be at least 2048-bit, and can be generated using a prime-finding algorithm
- The private key d should be generated by a true random number generator
- The total length of the message and signature $(x, (r, s))$ is about three times the length of the message x

ELGAMAL DIGITAL SIGNATURES: COMPUTATIONAL ASPECTS (2)

- Computing r requires exponentiation, achieveable by the square-and-multiply algorithm
- Computing s requires the inversion of k_E, which can be done using the extended Euclidean algorithm
- The ephemeral key k_E and r can be precomputed

ELGAMAL DIGITAL SIGNATURES: EXISTENTIAL FORGERY

An attacker can select integers i, j to compute a signature (r, s) such that it is valid for a message $x = si \bmod p - 1$:

Alice		**Oscar**		**Bob**
				$k_{pr} = d$ $k_{pub} = (p, \alpha, \beta)$
	$\xleftarrow{(p,\alpha,\beta)}$		$\xleftarrow{(p,\alpha,\beta)}$	
		1. select integers i, j where $\gcd(j, p-1) = 1$ 2. compute signature: $r \equiv \alpha^i \beta^j \bmod p$ $s \equiv -r\,j^{-1} \bmod p-1$ 3. compute message: $x \equiv s\,i \bmod p-1$		
	$\xleftarrow{(x,(r,s))}$			
verification: $t \equiv \beta^r \cdot r^s \bmod p$ since $t \equiv \alpha^x \bmod p$: valid signature!				

ELGAMAL DIGITAL SIGNATURES: EXISTENTIAL FORGERY (2)

- We can modify the signature scheme such that the cryptographic hash $h(x)$ of the message is signed, instead of the message x itself
- An attacker can then forge a signature for $h(x)$, but due to the one-way property of cryptographic hash functions it will be computationally infeasible to compute $x = h^{-1}(x)$
  - So this will not be "good enough" for the forgery to succeed, since the verification algorithm requires the original message x along with the signature (r, x)

THE DIGITAL SIGNATURE ALGORITHM (DSA)

- The original Elgamal signature algorithm is rarely used
- DSA is the most widely-used signature algorithm in practice
- DSA's main advantages over Elgamal are:
  - The signature is smaller
  - Some attacks against Elgamal are not applicable to DSA

DSA: KEY GENERATION

1. Choose a key length N and modulus length L.
2. Generate an N-bit prime p.
3. Find an L-bit prime divisor q of $p-1$.
4. Find an element α with $\mathrm{ord}(\alpha) = q$, i.e., α generates the subgroup with q elements.
5. Choose a random integer d with $0 < d < q$.
6. Compute $\beta \equiv \alpha^d \mod p$.

The public key is formed by $k_{pub} = (p, q, \alpha, \beta)$,
and the private key by $k_{pr} = d$.

DSA: PARAMETER LENGTHS

- DSA uses two cyclic groups: $\mathbb{Z}_p^*$ and a smaller subgroup of $\mathbb{Z}_p^*$
- NIST allows 112-bit security strength for protection up to the year 2030 (128-bit or higher is required for 2031 and beyond)

Security strength	(L, N) size (bits)	Signature size (bits)
112	(2048,224)	448
128	(3072,256)	512
192	(7680,384)	768
256	(15360,512)	1024

DSA SIGNATURE GENERATION

To compute the signature $\text{sig}_{k_{pr}}(x, k_E)$ for a message x:

1. Choose a random integer k_E such that $0 < k_E < q$.
2. Select an appropriate hash function $h(x)$ and compute the signature (r, s) as follows:

$$r \equiv (\alpha^{k_E} \bmod p) \bmod q,$$

$$s \equiv (h(x) + d \cdot r)k_E^{-1} \bmod q.$$

Just as with Elgamal, DSA becomes vulnerable if k_E is reused.

DSA SIGNATURE VERIFICATION

The receiver runs $ver_{k_{pub}}(x, (r, s))$ to verify the signature (r, s):

1. Compute auxiliary value $w \equiv s^{-1} \bmod q$.
2. Compute auxiliary value $u_1 \equiv w \cdot h(x) \bmod q$.
3. Compute auxiliary value $u_2 \equiv w \cdot r \bmod q$.
4. Compute $v \equiv (\alpha^{u_1} \cdot \beta^{u_2} \bmod p) \bmod q$.
5. If $v \equiv r \bmod q$, the signature is valid. Otherwise, it is invalid.

In other words, the verifier only accepts the signature if

$$(\alpha^{u_1} \cdot \beta^{u_2} \bmod p) \bmod q \equiv r \bmod q$$

DSA SIGNATURE VERIFICATION: PROOF

We start from the formula for computing s,

$$s \equiv (h(x) + dr)k_E^{-1} \bmod q,$$

which is equivalent to

$$k_E \equiv s^{-1}h(x) + ds^{-1}r \bmod q,$$

and substituting in u_1 and u_2 gives

$$k_E \equiv u_1 + du_2 \bmod q.$$

DSA SIGNATURE VERIFICATION: PROOF (2)

We raise α to both sides of the previous equivalence:

$$\alpha^{k_E} \equiv \alpha^{u_1+du_2} \mod p.$$

Since $\beta \equiv \alpha^d \mod p$, we can write:

$$\alpha^{k_E} \equiv \alpha^{u_1}\beta^{u_2} \mod p.$$

Reducing both sides of the equivalence to modulo q:

$$(\alpha^{k_E} \mod p) \mod q \equiv (\alpha^{u_1}\beta^{u_2} \mod p) \mod q.$$

DSA SIGNATURE VERIFICATION: PROOF (3)

Since r was constructed as $r \equiv (\alpha^{k_E} \bmod p) \bmod q$, substituting it into the previous equivalence yields

$$r \equiv (\alpha^{u_1} \beta^{u_2} \bmod p) \bmod q,$$

which is the expression used for signature verification that we were trying to prove.

DSA SIGNATURE VERIFICATION: EXAMPLE

Bob signs and sends x to Alice. The hash of x is $h(x) = 26$.

Alice | **Bob**

Bob

1. choose $p = 59$
2. choose $q = 29$
3. choose $\alpha = 3$
4. choose private key $d = 7$
5. $\beta = \alpha^d \equiv 4 \bmod 59$

$$\xleftarrow{(p,q,\alpha,\beta)=(59,29,3,4)}$$

sign:
compute hash of message $h(x) = 26$

1. choose ephemeral key $k_E = 10$
2. $r = (3^{10} \bmod 59) \equiv 20 \bmod 29$
3. $s = (26 + 7 \cdot 20) \cdot 3 \equiv 5 \bmod 29$

$$\xleftarrow{(x,(r,s))=(x,(20,5))}$$

Alice

verify:

1. $w = 5^{-1} \equiv 6 \bmod 29$
2. $u_1 = 6 \cdot 26 \equiv 11 \bmod 29$
3. $u_2 = 6 \cdot 20 \equiv 4 \bmod 29$
4. $v = (3^{11} \cdot 4^4 \bmod 59) \bmod 29 = 20$
5. $v \equiv r \bmod 29 \Longrightarrow$ valid signature

DSA: COMPUTATIONAL ASPECTS

- Computationally, the most demanding part is key generation
- The requirement is a cyclic group $\mathbb{Z}_p^*$ for a 2048-bit prime p that has a subgroup of order q, where q is a 224-bit prime
  - This is fulfilled if $p-1$ has a 224-bit prime factor q
- General approach is to first find the 224-bit prime q and then construct the larger prime p from it
- Signing is faster than Elgamal, since exponentiation is done to the power of k_E, where $k_E < q$
  - Verification is even faster

DSA: SECURITY

- In DSA, we choose the parameter lengths of p and q to protect against two different discrete logarithm attacks
- The length of p is chosen based on the index calculus attack, which is the fastest attack that can be used to compute the private key from the public key by solving $d = \log_{\alpha} \beta \bmod p$
  - So a 2048-bit p only offers 112-bit security
- The length of q is chosen based on a less powerful attack, since the index calculus attack is not applicable
  - So an N-bit q offers $\frac{N}{2}$-bit security
- So p and q should have equivalent security strength
  - As should the hash function (our next course topic)

ELLIPTIC-CURVE DSA

- The advantage of Elliptic-Curve DSA over DSA is similar to that of Elliptic-Curve DHKE over DHKE
- ECDSA is conceptually closely related to DSA, but is constructed in the group of an elliptic curve
  - The group operates on a set of points that are solutions to an equation representing an elliptic curve
- Due to the absence of strong attacks against elliptic curve cryptosystems, key lengths of 160-256 bits with ECDSA provide security equivalent to 1024-3072 bits with DSA