Advanced Topics in Cryptography

Lecture 8: Chosen-ciphertext security, the Cramer-Shoup cryptosystem. Bleichenbacher's attack against PKCS#1

Benny Pinkas

Related papers

– Lecture notes of Moni Naorhttp://www.cs.ioc.ee/yik/schools/win2004/naor-slides-2.5.ppt

– Lecture notes of Jonathan Katz http://www.cs.umd.edu/~jkatz/gradcrypto2/NOTES/lecture2.pdf

– Daniel Bleichenbacher, Chosen Ciphertext Attacks against Protocols Based on RSA Encryption Standard PKCS #1 http://www.bell-labs.com/user/bleichen/papers/pkcs.ps

Security against chosen-ciphertext attacks

- We show the public key to the adversary
- Adversary can ask to receive decryptions of messages of his choice
- Adversary chooses two messages m_0,m_1 (possibly based on the answers he previously received)
- Adversary is given an encryption $E(m_b)$, where $b \in_R \{0,1\}$
- Adversary can issue further decryption queries, but not $E(m_b)$
- Adversary guesses b

- Adversary succeeds if its probability of guessing b correctly is not negligibly close to ½

The Cramer-Shoup cryptosystem

- Cramer and Shoup suggested (in 1998) an encryption scheme which is practical and provably secure against chosen ciphertext attacks
- Security is based on the DDH assumption
- The overhead is only a few exponentiations

- The basic idea:
  - Add redundancy to the cryptosystem.
  - A ciphertext with the right redundancy is “valid”. Otherwise it is invalid.
  - Decryption is only performed for valid ciphertexts.

Cramer-Shoup “Lite” – security against “preprocessing” chosen ciphertext attacks

- Setup:
  - A subgroup G of order q, with generators g_1,g_2
- Key generation:
  - $x,y,a,b \leftarrow_R Z_q$
  - $h = (g_1)^x \cdot (g_2)^y \quad c = (g_1)^a \cdot (g_2)^b$
  - Public key = $\langle g_1,g_2,h,c \rangle$
  - Private key = $\langle x,y,a,b \rangle$
- Encryption of m:
  - $r \leftarrow_R Z_q$
  - Ciphertext is $\langle g_1^r, g_2^r, h^r \cdot m, c^r \rangle$
- Decryption of $\langle u,v,e,w \rangle$:
  - If $(w=u^a v^b)$ then output $e/(u^x v^y)$, otherwise no output.

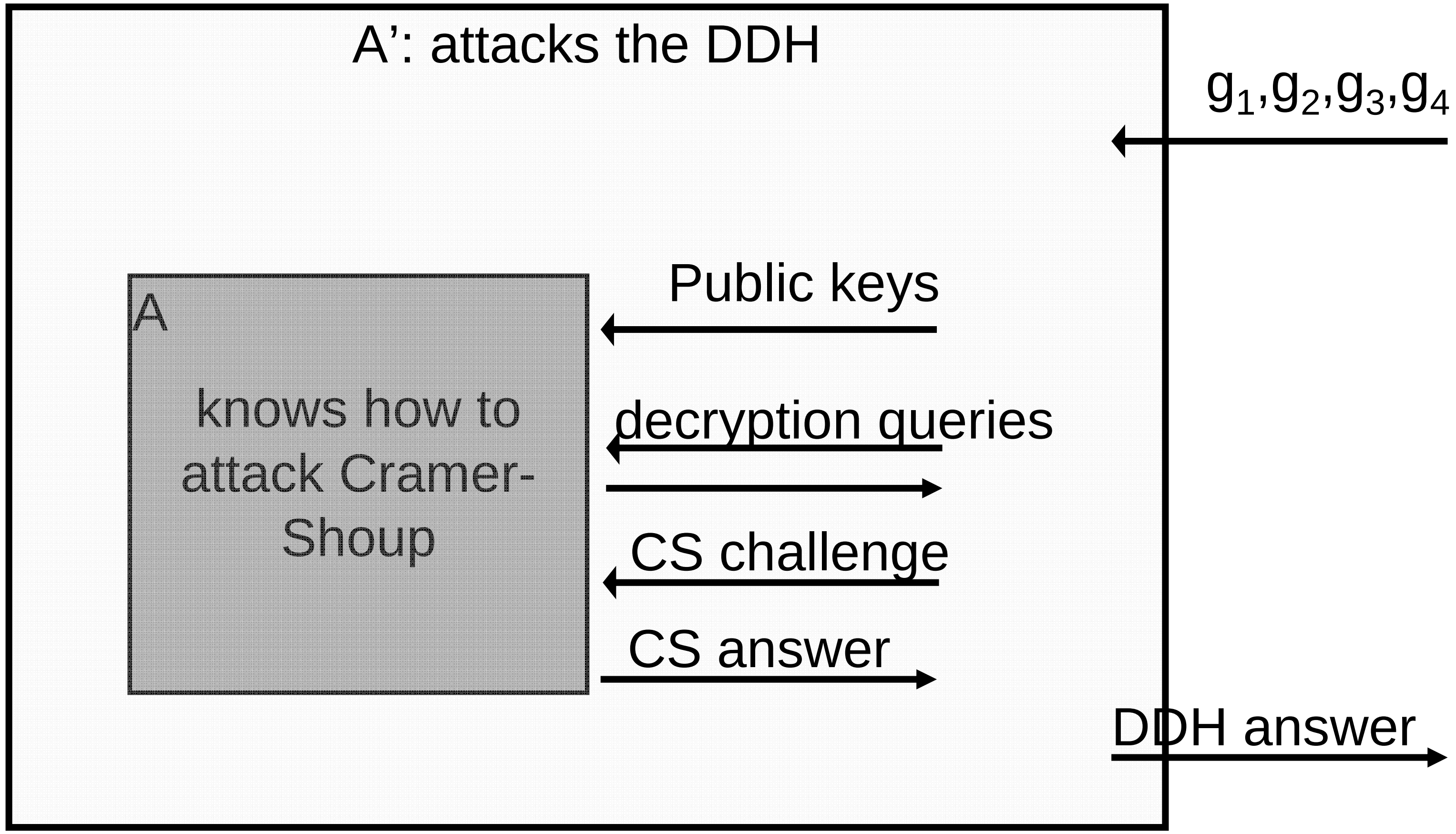
Correctness?

Overhead?

Security proof (against non-adaptive chosen ciphertext attacks)

- Assume that A attacks the cryptosystem. We build an A’ which breaks the DDH assumption.
- We are given an input to A’ and we generate a setting for A to work in. We want the following to hold:
  - If the input to A’ is a DDH tuple, then the setting of A is exactly as in the case it is attacking the cryptosystem.
  - If the input to A’ is a random tuple, then the setting of A provides it with an encryption of a random element.
  - The queries that A’ makes to the decryption oracle do not reveal anything.

Setting for the security proof



Cramer-Shoup Lite is insecure against chosen-ciphertext attacks which are based on the

- The ciphertext is $\langle u,v,e,w \rangle = \langle g_1^r, g_2^r, h^r \cdot m, c^r \rangle$
- The receiver checks that $u^a v^b = w$, but this check does not depend on $e = h^r \cdot m$.
- This enables attacks which are based on the homomorphic properties of the scheme.
  - The adversary can therefore change the challenge ciphertext $\langle g_1^r, g_2^r, h^r \cdot m_b, c^r \rangle$ to $\langle g_1^r, g_2^r, h^r \cdot m \cdot \alpha, c^r \rangle$, send it as a decryption query and obtain $m \cdot \alpha$.
  - Alternatively, it could generate the ciphertext $\langle g_1^{rr'}, g_2^{rr'}, h^{rr'} \cdot m_b^{r'}, c^{rr'} \rangle$, send it as a decryption query and learn $m^{r'}$.

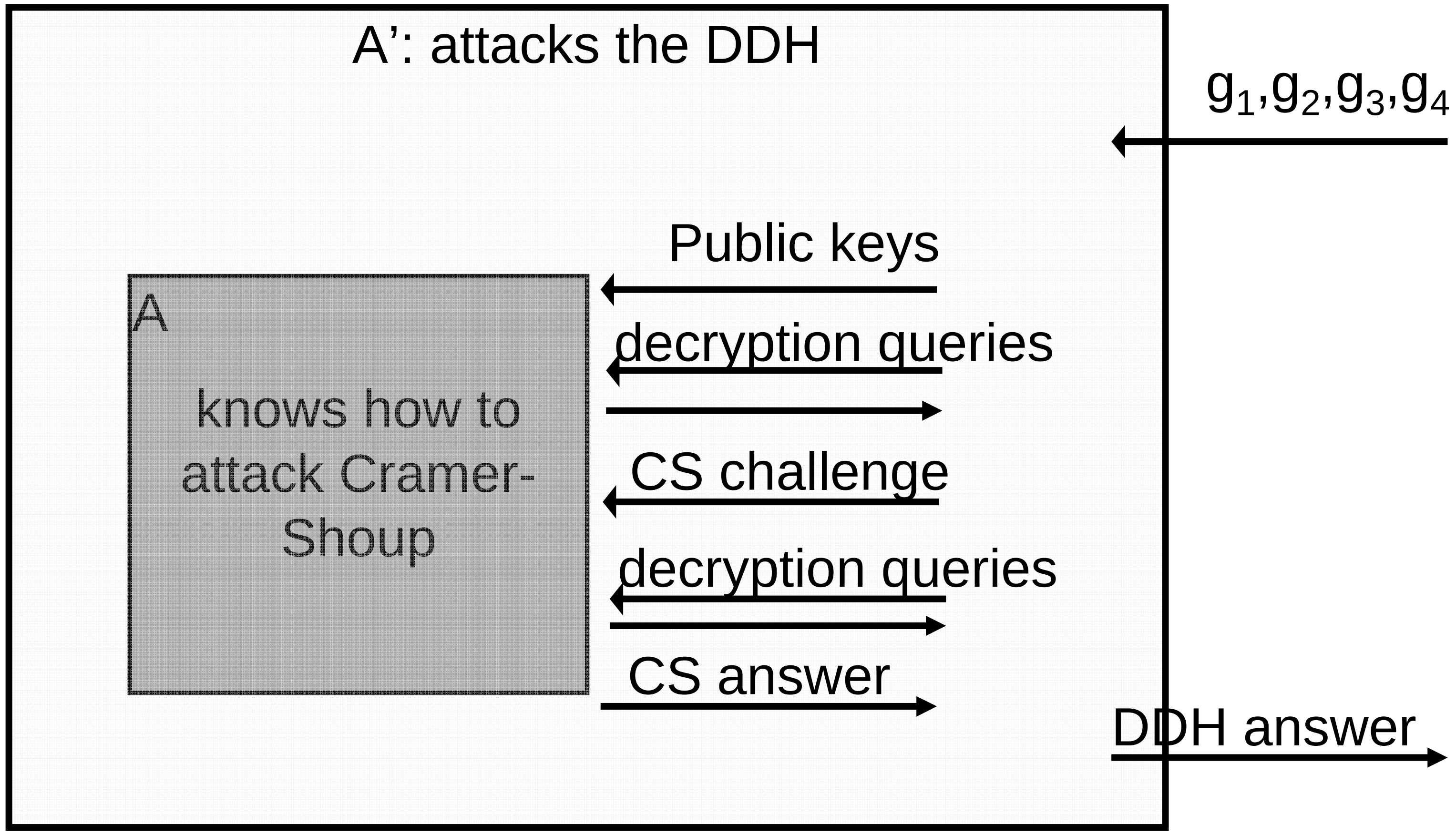
The Cramer-Shoup cryptosystem

- Key generation:
  - $x,y,a,b,\underline{a'},\underline{b'} \leftarrow_R Z_q$
  - $h = (g_1)^x \cdot (g_2)^y \quad c = (g_1)^a \cdot (g_2)^b \quad \underline{d} = (g_1)^{a'} \cdot (g_2)^{b'}$
  - H is a collision resistant hash function
  - Public key = $\langle g_1, g_2, h, c, d, H \rangle$
  - Private key = $\langle x, y, a, b, a', b' \rangle$
- Encryption of m:
  - $r \leftarrow_R Z_q$
  - Ciphertext is $\langle g_1^r, g_2^r, h^r \cdot m, \underline{(cd^\alpha)^r} \rangle$, $\alpha = H(g_1^r, g_2^r, h^r \cdot m)$
- Decryption of $\langle u,v,e,w \rangle$:
  - If $(w = u^{a+\alpha a'} \cdot v^{b+\alpha b'})$, where $\alpha = H(u,v,e)$, then output $e/(u^x v^y)$, otherwise no output.
- Why can't we remove d, and let the last element be $c^{\alpha r}$?

Security proof (against adaptive chosen ciphertext attacks)

- Assume that A attacks the cryptosystem. We build an A’ which breaks the DDH assumption.
- We are given an input to A’ and we generate a setting for A to work in. We want the following to hold:
  - If the input to A’ is a DDH tuple, then the setting of A is exactly as in the case it is attacking the cryptosystem.
  - If the input to A’ is a random tuple, then the setting of A provides it with an encryption of a random element.
  - The queries that A’ makes to the decryption oracle do not reveal anything.
  - A’ can make adaptive decryption queries, even after seeing the challenge ciphertext.

Setting for the security proof



Constructing A'

- Our input is (g_1,g_2,g_3,g_4), which is either a DDH tuple (of the form g,g^a,g^b,g^{ab}, namely $\log_{g1}(g_3)=\log_{g2}(g_4)$), or a random tuple.
  - $x,y,a,b,a',b' \leftarrow_R Z_q$
  - $h = (g_1)^x \cdot (g_2)^y$ $\quad c = (g_1)^a \cdot (g_2)^b$ $\quad d = (g_1)^{a'} \cdot (g_2)^{b'}$
  - Public key = $\langle g_1,g_2,h,c,d,H \rangle$
  - Private key = $\langle x,y,a,b,a',b' \rangle$
  - Answer decryption queries of A, and then receive m_0,m_1.
  - Choose $s \in_R \{0,1\}$.
  - Send to A the ciphertext $\langle g_3, g_4, g_3^x g_4^y \cdot m_s, g_3^{a+\alpha a'} g_4^{b+\alpha b'} \rangle$
  - Answer further decryption queries of A
  - If the response of A is equal to s then output "DDH tuple", otherwise output "random tuple"

Case 1: The input of A’ is a DDH tuple

- THM: If A’ receives an input which is a DDH tuple, then the view of A is the same as when it is interacting with a real cryptosystem.
- Corollary: Pr(A’ outputs “DDH” | DDH input) = Pr(A succeeds when attacking a real cryptosystem)
- Proof:
  – The same as in the Cramer-Shoup Lite case.

Case 2: The input of A’ is a random tuple

- THM: If A’ receives an input which is a random tuple, then (except with negligible probability) A has no information about the bit *s* chosen by A’.

  Namely, | Pr(A guesses s | random tuple) – ½ | is negligible.

- Corollary:
  - | Pr(A’ outputs “DDH” | random tuple input) – ½ | = | Pr(A guesses s | random tuple) – ½ |, and is negligible
  - | Pr(A’ outputs “DDH” | DDH input) – Pr(A’ outputs “DDH” | random tuple input) |

    = |Pr(A succeeds when attacking a real cryptosystem) - ½|

Proof of Theorem 2

- As before, we prove the theorem even for the case of an adversary which can compute discrete logarithms.
  - The adversary A therefore knows $\gamma=\log_{g1}g_2$
- The proof shows that any "bad" decryption query ($\langle u,v,e,w \rangle$ in which $\log_{g1}u \neq \log_{g2}w$), is rejected (except with negligible probability).
- Therefore A can only make "good" queries, and exactly as in the "Lite" case, we show that A learns nothing from these queries.

Proof of the theorem

- Claim 1: With all but negligible probability, all “bad” decryption queries (u,v,e,w) s.t. $\log_{g_1} u \neq \log_{g_2} v$, fail.
- Proof:
  - A, which can compute discrete logs, knows that
    - $\log_{g_1} c = a+b\cdot\gamma$
    - $\log_{g_1} d = a'+b'\cdot\gamma$
    - $\log_{g_1} w^* = (a+\alpha a')r+(b+\alpha b')\gamma r'$. This is revealed from the challenge query $\langle g_3=g_1^{r}, g_4=g_1^{r'}, e^*=g_3^{x} g_4^{y}\cdot m_s, w^*=g_3^{a+\alpha a'} g_4^{b+\alpha b'}\rangle$
  - A cannot submit the query $\langle g_3,g_4,e^*,w^*\rangle$
  - If it submits a query $\langle u,v,e,w\rangle$ such that $(u,v,e)=(g_3,g_4,e^*)$ but $w\neq w^*$, then the query is always rejected.

Proof (contd)

- (u,v,e) must therefore be different from (g_3,g_4,e^*)
- If $H(u,v,e)=H(g_3,g_4,e^*)$, then A breaks the collision intractability of H…
- Therefore $\alpha' = H(u,v,e) \neq H(g_3,g_4,e^*) = \alpha$.
- The decryption query of A is accepted only if $\log_{g1} w = (a+\alpha' a')\cdot\log_{g1} u + (b+\alpha' b')\cdot\gamma\cdot\log_{g2} v$, but this equation is independent of the known equations
    - $\log_{g1} c = a+b\cdot\gamma$
    - $\log_{g1} d = a'+b'\cdot\gamma$
    - $\log_{g1} w^* = (a+\alpha a')r+(b+\alpha b')\gamma r'$
  - (This is true since $r \neq r'$, $\log_{g1} u \neq \log_{g2} v$, and $\alpha \neq \alpha'$.)
  - Therefore A generates a good query with probability 1/q.

Proof of the theorem (contd, exactly as in the "Lite" case)

- Claim 2: Assuming all "bad" decryption queries are rejected, A learns no information about *x* and *y.*
- Proof:
  - A knows $\gamma = \log_{g1} g_2$. The public key contains $h = g_1^x g_2^y$, and A therefore learns that $\log_{g1} h = x + y \cdot \gamma$.
  - Bad (rejected) queries reveal nothing about (x,y), since the rejection is based on the values of (a,b) alone.
  - For good queries (u,v,e,w), A learns $e/m = g_1^{rx} g_2^{ry}$. Namely, that $\log_{g1}(e/m) = xr + yr \cdot \gamma$. (Which is a relation it already knows.)
- Claims 1+ 2 $\rightarrow$ after n queries, with probability 1-n/q it holds that the ciphertext $\langle g_3, g_4, g_3^x g_4^y \cdot m_s, g_3^a g_4^b \rangle$ has (q-n) equal probability options for (x,y), and therfore for m.
- QED

Chosen Ciphertext Attacks against Protocols Based on RSA Encryption Standard PKCS #1

- Is the chosen-ciphertext attack scenario reasonable?
  - Who is stupid enough to answer decryption queries?
- The following attack does not require decryption of chosen ciphertexts, but rather only learning a single bit about the encrypted plaintext.
- PKCS #1 protocol
  - A protocol of encrypting plaintexts in RSA
  - Describes how to pad messages so that they fit in Z_n^*
  - If a server which uses PKCS #1 receives a ciphertext which decrypts to a message that does not conform with the protocol, an error message is returned.
  - A reply which includes an error message provides a single bit of information about the plaintext.

PKCS #1

- An RSA encryption standard.
- Let *n* be the RSA modulus, and *k* be the byte length of *n*.
- An encryption of a data block D, of |D| bytes, first generates a string x of the following form:
  - The most significant byte is 00
  - The next byte is 02
  - They are followed by a padding string of at least 8 nonzero bytes.
  - The next byte is 0.
  - The rest of the bytes include D. (Therefore $|D| \leq k-11$ bytes, and the length of the padding string is $k-3-|D|$.)
- The string x is encrypted. The ciphertext is x^e mod n.

The attack

- The attacker is given c, and wants to find $m=c^d$.
- The attacker picks random numbers s, and asks to decrypt $c'=c\cdot s^e \bmod n$. If it does not receive an error message it concludes that $m\cdot s$ is PKCS compliant.
  - Let $B=2^{8(k-2)}$
  - Then if $m\cdot s$ is PKCS compliant, then
    - $2B \le m\cdot s \bmod n \le 3B-1$

The attack

- Given *c*, choose random integers s_0, until $c(s_0)^e$ is a valid ciphertext. Then,
  - $c_0 = c\cdot (s_0)^e \bmod n$.
  - $M_0 = \{ [2B,3B-1] \}$. (We know that m_0 is in [2B,3B-1].)
  - In general, $M_i=\{[a_j,b_j]\}$, s.t. $\exists$ j for which $m_0 \in [a_j,b_j]$.
- Then, for i=1,…
  - Look for s_i, such that $c_0(s_i)^e$ is PKCS compliant.
  - $\exists$ r such that $2B \le m_0 s_i - r\cdot n \le 3B-1$, and therefore

$$\frac{2B + r\cdot n}{s_i} \le m_0 \le \frac{3B-1+r\cdot n}{s_i} \quad (1)$$

  - If $m_0 \in [a_j,b_j]$, then we also get that $a_j s_i-(3B-1) \le r\cdot n \le b_j s_i-2B$
  - This provides possible values for r, which together with (1) define new, smaller, ranges for m_0.

Analysis

- Heuristic arguments show that the number of intervals is step i is bounded by

$$1 + 2^{i-1} \cdot s_i \cdot \left(\frac{B}{n}\right)^i$$

  and therefore decreases exponentially.

- The analysis shows that for a a 1024 bit RSA modulus, about 2^{20} decryption attempts are needed. (This was verified by experiments.)
  – Is this reasonable?

Applying the attack

- A human user is unlikely to answer to 2^{20} incorrect messages
- But a computer will
- Many implementations of SSL V3.0 return an error message if RSA is chosen as the encryption function, and the encrypted message is not PKCS compliant.
  - In fact, the SSL protocol requires the message to comply with an additional integrity check. But some implementations of the protocol return an answer even if the message complies with PKCS and does not pass the integrity test.