Cryptography I
Exercise sheet 7

Ilja Kuzovkin

November 3, 2010

1 Exercise 1: Rabin identification scheme

1.1 The scheme

We have to "people": Verifier (V), who wants to make sure, that Prover (P) is the Prover indeed. For that Prover must demonstrate knowledge of a secret component by solving the challenge Verifier will give him. In our case secret component is pair of Blum primes p and q and challenge is finding square roots of $C \ mod \ n$, where $n = pq$.

1. Prover generates two Blum primes (congruent to 3 modulo 4) p and q
2. Prover calculates $n = pq$
3. Prover generates random number R
4. Prover **sends** R and n to Verifier
5. Verifier generates random number S
6. Verifier computes $C = (RS)^2 \ mod \ n$
7. Verifier **sends** C to the Prover
8. Prover finds 4 square roots of C using the idea, which is the basis of the Rabin's cryptosystem decryption process
   (a) Using Euclid's algorithm find h and k such that $hp + kq = 1$
   (b) Find $a = C^{\frac{(p+1)}{4}} \ mod \ p$
   (c) Find $b = C^{\frac{(p+1)}{4}} \ mod \ q$
   (d) Find $x = (hpb + kqa) \ mod \ n$
   (e) Find $y = (hpb - kqa) \ mod \ n$
   (f) Square roots of C are $\{x, -x, y, -y\}$
9. Prover **sends** roots $\{x, -x, y, -y\}$ to the Verifier
10. Verifer checks that $\exists i \ s_i \in$ {x, -x, y, -y} : $s_i = RS$. This will demostrate, that Prover was able to find correct square roots of C and knows the secret (p, q).

1.2 Is it secure?

1. We know that breaking Rabin's cryptosystem is as hard as factoring n, so cryptological primitive used in our scheme is secure.
2. As far as I can see Zero-Knowledge proof holds in this case: only information Verifier is receiving are square roots, which, as we see from the Rabin's cryptosystem decryption scheme give Verifier very small information.
3. There is extremely small probability that intruder will be able to guess square roots
4. If Verifier is evil and will try $S = 0$ and $S = 1$ to get some information, he still will get nothing, because R is also randomly generated

It seems to me, that such a identification scheme is quite secure, at least at a first glance.

2 Exercise 2: RSA compression function

2.1 One-wayness

To see if hash function $h(x) = g^x \ mod \ n$ is one-way function lets play it through and see what we might need to break it. Lets assume we have message $m < n$, we make hash of it $y = h(m)$, so $y = g^x \ mod \ n$. The guy who want to break it knows y, g and n and he wants to find x. This is classical discrete logarithm problem, which is known to be hard[1]. So we cannot easily find x and one-wayness holds.

2.2 2nd preimage resistant

Lets assume $G = \{0, 1, 2, 3, 4, 5, 6\}$, then $n = 7$ and $g = 3$ is the generator. We want to encode message $m = 4$, so we compute $y = h(m) = g^m \ mod \ n = 3^4 \ mod \ 7 = 4$. But now we cat get same y if instead of $m = 4$ we use $m = 4 + k(n - 1)$, where $k \in \mathbb{N}$, we know that $m \in \mathbb{Z}_n^2$, so the last m we can encode is 48:
$3^4 \ mod \ 7 = 4$
$3^{4+(n-1)} \ mod \ 7 = 3^{4+6} \ mod \ 7 = 3^{10} \ mod \ 7 = 4$
$3^{4+2(n-1)} \ mod \ 7 = 3^{4+12} \ mod \ 7 = 3^{16} \ mod \ 7 = 4$
...
$3^{4+7(n-1)} \ mod \ n = 3^{4+42} \ mod \ n = 3^{46} \ mod \ 7 = 4$
So this hash function is not 2nd preimage resistant.

2.3 Collision resistant

Because our hash function is not preimage resistant it is not collision resistant also. One can take any suitable m, compute $y = h(m)$ and find $m' \neq m$ such that $h(m) = h(m')$ using techinque described in section 2.2 of this document.

3 Exercise 3: E-voting with RSA

3.1 Secure way of encrypting single bit message

Lets say our RSA modulo n is 1024 bit long. Then, to encrypt single $bit = \{0, 1\}$ we add to it some random number c and special condition is, that k-th bit of c is known to be 0. So our message will be $m = bit + c$ and we encrypt it.

```
                * - this is our k-th bit
c   = <...>101100101101<...>
bit =           1
----------------------------
m   = <...>101101101101<...>
```

After decryption we look at k-th bit of m. If it is still 0, then the vote was "against" and if it is "1", then the vote was "for".

3.2 3 wrong things about system described in the task 7.3.2

Lets see an example using toy values. Let RSA be $n = 77$, $e = 7$, $d = 43$ (those numbers are suitable for RSA encryption and decryption). Lets say we have 20 "for" votes and 18 "against" votes. "for" vote encrypted will be

$$c_f = 2^7 \ mod \ 77 = 51$$

and "against"

$$c_a = 1^7 \ mod \ 77 = 1$$

The product of all encryptions

$$c_r = \prod_{i=1}^{n} c_i = 51^{20} * 1^{18} = 14171098670753043575626125424226001$$

If we try to decrypt c_r we will get

$$c_r^{43} \ mod \ 77 = 67$$

which is equal to $(2^{20})^7 \ mod \ 77$, so we can find that there were 20 "for" votes.

3.2.1 If voter will encrypt not 1 or 2

If voter will encrypt and send "4" then central server will say that there was 21 "for" vote.
If voter will encrypt and send "0" then we will get 0 as total result.

3.2.2 It is infeasible for countries with large population

If we will perform this voting in country with large population then a in the equation $(2^a)^{public\_key\_e} \ mod \ n$ will be too big and computation will be infeasible to complete.

3.2.3 1 is always 1

Even if we encrypt 1 it still be 1 and eavesdropping adversary will be able to distinguish "for" and "against" votes.

3.3 Improved scheme for 10-people voting

This scheme is based on the fact that we need only 10 people to give their votes.

1. Voter picks random number k from range $2^{10} + 1 \leq k \leq n - 1$
2. Voter substiutes 11 last bits of this number by zeros $k = ...00000000000$
3. If he votes "for" he puts 2 in the end of k, so that $k = ...00000000010$
4. If he votes "for" he puts 1 in the end of k, so that $k = ...00000000001$
5. All 10 voters encrypt their messages and send them to the server

6. Server calculates $c_r = \prod_{i=1}^{10} c_i$

7. Server decrypts $m = dec(c_r)$

8. The last 11 bits of m represent $2^{amount\_of\_for\_votes}$.
   If, for example, $m = ...00001000000$ then we know, that $1000000_{bin} = 64_{dec} = 2^6$, so there were 6 "for" votes.

3.4 Flaws in scheme described in section 3.3

1. I think that the knowledge, that 11 last bit of the message, which voter will send can be 00000000001 or 00000000002 can somehow be used by attacker. But I can't tell how exactly.

2. Voter still can give other answer than 1 or 2 and spoil results

3.5 Use of zero-knowledge proof in the voting system

1. It can be used to make sure voter is sending correct vote, in correct format.[2]

2. If we know the right answer from another source we can check if Server has the same answer without sending it over the network and revealing the right answer to the Server

References

[1] http://en.wikipedia.org/wiki/Discrete_logarithm#Algorithms

[2] http://www.brics.dk/~jg/ACNS05VoteProofFull.pdf