Attacking Cryptographic Schemes Based on 'Perturbation Polynomials'

Martin Albrecht (Royal Holloway),
Craig Gentry (IBM), Shai Halevi (IBM),
Jonathan Katz (Univ. of MD)

The moral

- Implementing secure protocols in MANETs/ sensor-networks can be challenging
  - Low bandwidth, memory, computational power
  - Limited battery life
- Much work designing new and highly efficient protocols tailored to this setting
- Sometimes, rigorous security analysis sacrificed for better efficiency
  - Replaced with heuristic analysis

This is a bad idea!

Outline of the talk

- Key predistribution
- An optimal, information-theoretic scheme
- A modified scheme by Zhang et al.
- Attacking the modified scheme
- Extensions and conclusions

Key predistribution

- Goal: distribute keying material to N nodes, so each pair can compute a shared key
  - Off-line key-predistribution
  - On-line computation of shared keys
- Two trivial solutions:
  - One key shared by all nodes
    - Compromise of one node compromises entire network
  - Independent key shared by each pair of nodes
    - O(N) storage per node
- A not-so-trivial solution [Sakai et al. 2000]:
  - Identity-based key agreement
    - O(1) storage, full resilience
    - But expensive computation (pairing)

‘Optimal’ storage/resilience tradeoff

- [Blom ‘84], [Blundo et al. ‘98]
- These schemes guarantee the following:
  - Any pair of nodes shares a key
  - A key shared by uncompromised nodes is *information-theoretically* secret
  - As long as t or fewer nodes are compromised
- Storage $O(t)$ per node
  - This is *optimal* for schemes satisfying the above
- Computation is “cheap”
  - No “public key operations”

The scheme of Blundo et al.

- Choose a random symmetric polynomial *$F(x,y)$* of degree *t* in each variable
  - $F(x,y)= F(y,x)$
- Node *i* given coefficients of (univariate) polynomial $s_i(y) = F(i,y)$
- Key shared by *i* and *j* is $s_i(j) = F(i,j) = s_j(i)$

- *After compromising $t+1$ nodes, attacker can recover $F(x,y)$ by interpolation*

Better than Blundo?

- If t large, even $O(t)$ storage is expensive
- Can we do better?
  - E.g., by giving up info-theoretic security
  - Without paying in expensive operations?

Perturbation polynomial

- [Zhang et al., MobiHoc '07]
  - Other variations by Zhang et al. (INFOCOM '08), Subramanian et al. (PerCom '07)
- Basic idea:
  - Give node *i* a polynomial $s_i(y)$ that is "close", but not equal, to $F(i,y)$
  - Nodes *i* and *j* generate a shared key using the high-order bits of $s_i(j)$, $s_j(i)$, respectively
  - Harder(?) for an adversary to recover $F(x,y)$, even after compromising many nodes

The scheme of Zhang et al.

- p a prime, $r < p$ a "noise parameter"
- Choose random symmetric F(x,y) as before
- Choose random degree-t univariate $g(y), h(y)$
  - Find i's such that both g(i) and h(i) are small
    SMALL = {i : 0 ≤ g(i), h(i) ≤ r} (mod p)
- For i ∈ SMALL, choose random b ← {0,1}
  - Node is given "name" i and coefficients of

$$s_i(y) = F(i,y) + g(y) \qquad \text{if } b = 0$$
$$s_i(y) = F(i,y) + h(y) \qquad \text{if } b = 1$$

- $|s_i(j) - s_j(i)| \le r$ for any $i, j \in$ SMALL
  - Nodes i, j agree on a shared key using high-order bits

Suggested parameters

- $p \approx 2^{32}$, $r \approx 2^{22}$, $t=76$
- Number of bits in key = $\log(p/r) = 10$
  - Run scheme many times for more key bits
- Storage per node: $(t+1) \log p \approx 2460$ bits
- Storage per key bit ≈ 246 bits
- Blundo scheme with this much storage is resilient to ≈ 246 corruptions
- Zhang et al. claim resistance against arbitrarily many corruptions

"Warm-up attack" using list decoding

- Compromise n=4t+1 nodes
  - Learn coefficients of $s_1(y),\ldots, s_n(y)$
- For any victim j^*, set $y_i = s_i(j^*)$
- Note: $y_i \in \{f_0(i), f_1(i)\}$
  - $f_0(y) = F(y,j^*)+g(j^*)$, $f_1(y) = F(y,j^*)+h(j^*)$
- For some *b,* more than half the y_i's $= f_b(i)$
  - Use *list decoding* to recover this $f_b(y)$
    - Algorithm of [Ar et al. 1998]
  - Compute shared key between j* and any i*
    - $s_{j*}(i^*) \approx f_b(i^*)$

The "real attack"

- Breaks "generalized" version of scheme with more noise:
  - $s_i(y) = F(i,y) + \alpha_i\, g(y) + \beta_i\, h(y)$
  - Small α_i, $\beta_i \in [-u, u]$
- Only needs to corrupt t+3 nodes
- Takes time $O(t^3 + t\, u^3)$
  - Note: u cannot be too large, to share even a 1-bit key we need $4ur < p$
  - Attack is faster than key setup

Implementation

- Attack implemented on a desktop PC

p	r	t	setup time	attack time
$2^{32}-5$	2^{22}	76	60 min	10 min
$2^{36}-5$	2^{24}	77	1060 min	8 min

It takes a long time to compute the set SMALL = $\{i : 0 \le g(i), h(i) \le r\}$

The info-theoretic protection

Noise dimension

- The "noise space" is spanned by g(), h()
  - Two dimensional space, can be identified after corrupting (t+1)+2 = t+3 nodes
- For i ∈ SMALL, g(i), h(i) are small
  - Use lattice-reduction to find g(), h()
  - Low-dimensional noise-space
    ➔ only need to reduce lattices of low dimension
    - Dimension < 20 for the suggested parameters
- Once g(), h() are found, easy to recover the master polynomial *F(x,y)*

Step 1: identify the noise space

- Corrupt n=t+3 nodes, get

$$\mathbf{s}_i = \mathbf{f}_i + \alpha_i\, \mathbf{g} + \beta_i\, \mathbf{h}$$

- We know

$$\mathbf{f}_{t+1} = \Sigma_{i=0\ldots t}\, \lambda_i\, \mathbf{f}_i \text{ and } \mathbf{f}_{t+2} = \Sigma_{i=0\ldots t}\, \lambda'_i\, \mathbf{f}_i$$

- So: $\mathbf{v} = \mathbf{s}_{t+1} - \Sigma_{i=0\ldots t}\, \lambda_i\, \mathbf{s}_i \in \text{span}(\mathbf{g}, \mathbf{h})$

  $\mathbf{v'} = \mathbf{s}_{t+2} - \Sigma_{i=0\ldots t}\, \lambda'_i\, \mathbf{s}_i \in \text{span}(\mathbf{g}, \mathbf{h})$

- **v,v'** likely to be linearly independent
  - Likely to be a basis for span(**g**, **h**)!

Step 2: find g and h

- We have **v**, **v'** s.t. span(**v**,**v'**) = span(**g**,**h**)
- Find g, h using the fact that g(id), h(id) are "small" modulo p
- To do this, find short vectors in the lattice:

$$\begin{pmatrix} v(x_1) & v(x_2) & \cdots & v(x_k) \\ v'(x_1) & v'(x_2) & \cdots & v'(x_k) \\ p & 0 & \cdots & 0 \\ 0 & p & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & p \end{pmatrix}$$

k can be small
($k < 20$)

Step 3: find F

- F is symmetric, so for all i, j
  $s_i(j) - \alpha_i g(j) - \beta_i h(j) = s_j(i) - \alpha_j g(i) - \beta_j h(i)$
  - Gives $O(n^2)$ equations in 2n unknowns (α_i, β_i)
  - But under-determined!
    - Exactly 3 degrees of freedom
- Exhaustive search for three of the α_i, β_i in [-u, u]
  - Total time $O(t^3 + t\ u^3)$
  - Or use lattices to do it even faster..

Other Perturbation Polynomial Schemes

- Authentication scheme by Zhang et al. from INFOCOM 2008
- Access-control scheme by Subramanian et al. from PerCom 2007
- The same type of attacks apply there too
  - Attacks are actually easier

Conclusions

The ‘perturbation polynomials’ approach is dead

Moral: rigorous security analysis is *crucial*

Thank you!