Attacking Cryptographic Schemes Based on "Perturbation Polynomials"

Martin Albrecht[*] Craig Gentry[†] Shai Halevi[‡] Jonathan Katz[§]

Abstract

We show attacks on several cryptographic schemes that have recently been proposed for achieving various security goals in sensor networks. Roughly speaking, these schemes all use "perturbation polynomials" to add "noise" to polynomial-based systems that offer information-theoretic security, in an attempt to increase the resilience threshold while maintaining efficiency. We show that the heuristic security arguments given for these modified schemes do not hold, and that they can be completely broken once we allow even a slight extension of the parameters beyond those achieved by the underlying information-theoretic schemes.

Our attacks apply to the key predistribution scheme of Zhang et al. (MobiHoc 2007), the access-control schemes of Subramanian et al. (PerCom 2007), and the authentication schemes of Zhang et al. (INFOCOM 2008).

1 Introduction

Implementing standard security mechanisms in sensor networks is often challenging due to the constrained nature of sensor nodes: they have limited battery life, relatively low computational power, and limited memory. As such, a significant body of research has focused on the design of special-purpose, highly efficient cryptographic schemes for sensor network applications.

Here, we examine an approach based on "perturbation polynomials" that has been used to construct several recent schemes [7, 5, 6]. This approach, initiated by Zhang, Tran, Zhu, and Cao [7] and Subramanian, Yang, and Zhang [5], takes a polynomial-based scheme that offers *information-theoretic* (i.e., perfect) security for some "resilience parameter" t — e.g., a bound on the number of compromised nodes or the number of messages authenticated — and then modifies this underlying scheme so that the resilience is supposedly increased against a *computationally bounded* attacker. The common idea is to add a small amount of "noise" to the low-degree polynomials used in the original scheme; the claim is that the presence of this noise makes breaking the scheme infeasible even in regimes well beyond the original resilience parameter. Unfortunately, we show here that this naive view is unfounded.

[*]Royal Holloway University. Work supported by a Royal Holloway Valerie Myerscough scholarship and NSF grant #0821725. Email: `M.R.Albrecht@rhul.ac.uk`

[†]Stanford University. Work done while visiting IBM. Email: `cgentry@cs.stanford.edu`

[‡]IBM T.J. Watson Research Center. Work supported by the US Army Research Laboratory and the UK Ministry of Defence under agreement number W911NF-06-3-0001. Email: `shaih@alum.mit.edu`

[§]University of Maryland. Work done while visiting IBM. Work supported by NSF Trusted Computing grant #0627306, the U.S. DoD/ARO MURI program, and the US Army Research Laboratory and the UK Ministry of Defence under agreement number W911NF-06-3-0001. Email: `jkatz@cs.umd.edu`

We describe efficient attacks against the schemes from [7, 5, 6], demonstrating that these scheme do not offer any better resilience than the original, information-theoretic schemes on which they are based. We provide theoretical justification as to why our attacks work, as well as experimental evidence that convincingly illustrates their effectiveness. Our results cast strong doubt on the viability of the "perturbation polynomials" approach for the design of secure cryptographic schemes.

1.1 Organization of the Paper

We focus the bulk of our attention on the initial paper of Zhang et al. [7], which concerns key predistribution in sensor networks. A description of their scheme, and details of our attack, are given in Section 2. In Section 3 we show how to apply our attack to a set of message authentication schemes suggested by Zhang et al. [6], and in Section 4 we show the same for a system for secure data storage/retrieval proposed by Subramanian et al. [5].

2 The Key Predistribution Scheme of Zhang et al.

2.1 Background

Schemes for *key predistribution* enable nodes in a large network to agree on pairwise secret keys. Before deployment, a central authority loads some secret information s_i onto each node i, for $i \in \{1, \ldots, N\}$ (where N is the network size). Later, any two nodes i and j can agree on a shared key $k_{i,j}$ of length κ using their respective secret information. (Probabilistic schemes, where two nodes are only able to compute a shared key with high probability, have also been considered but will not concern us here.) The security goal is to offer resilience as large as possible, where a scheme has resilience t if an adversary who compromises t nodes $I = \{i_1, \ldots, i_t\}$ is still unable to derive any information about the shared key $k_{i,j}$ for any i, j such that $i, j \notin I$. Efficiency considerations require computation of the shared keys to be fast, thus ruling out standard public-key approaches, and dictate that the storage (i.e., the size of the keying information s_i) should be minimized.

One simple approach is for all nodes to share a single key k (i.e., set $s_i = k$ for all i) that is used also as the pairwise key for any pair of nodes. While having minimal storage, this scheme has resilience $t = 0$ since it is completely broken after only one node is compromised. A second trivial approach is for each pair of nodes to store an independent key. This has optimal resilience $t = N$, but the storage requirement of $\binom{N}{2} \cdot \kappa$ is unacceptably high.

Blundo et al. [3] show that resilience t requires storage $(t+1) \cdot \kappa$ if information-theoretic security is desired; Blom [2] and Blundo et al. show schemes meeting this bound. Let $\mathbb{F}$ be a field whose elements can be used as pairwise keys. To achieve resilience t using the scheme of Blundo et al., the authority chooses a random symmetric, bivariate polynomial $F \in \mathbb{F}[x, y]$ of degree t in each variable as the master secret key; a node with identity $i \in \mathbb{F}$ is given the univariate polynomial $s_i(y) = F(i, y)$ as its secret information. The shared key $k_{i,j}$ between nodes i, j is $s_i(j) = F(i, j) = s_j(i)$, which both parties can compute (using the fact that F is symmetric). It is not hard to see that an attacker who compromises at most t nodes learns no information about any key that is shared between non-compromised nodes. However, an attacker who compromises $t + 1$ nodes can use interpolation to recover the master polynomial and thus recover all the keys in the system.

2.2 The Scheme of Zhang et al.

Zhang et al. [7] suggested a "noisy" version of the above scheme, and claimed that the new scheme has improved resilience for some fixed amount of storage. Roughly, their idea is to give node i a polynomial $s_i(y)$ that is "close", but not exactly equal, to $F(i, y)$. Nodes i and j can compute $s_i(j)$ and $s_j(i)$ as before; these results will no longer be equal, but because they are close they can still be used to derive a shared key (by, e.g., using the high-order bits). The hope was that the addition of noise to the nodes' secret information would prevent reconstruction of the master secret F even if an adversary corrupts many more than $t+1$ nodes; in fact, Zhang et al. claim optimal resilience $t = N$ as long as the adversary is computationally bounded. (Of course, for a computationally unbounded adversary the lower bound from [3] applies.) We show that this is not the case.

We first describe their scheme in further detail. Let p be a prime, and let $r < p$ be a "noise bound". Elements in $\mathbb{Z}_p$ are represented as integers in $[0, p-1]$ in the natural way and we freely interchange between the two representations (so, e.g., $a < b$ means that the integer representation of a is smaller than the integer representation of b). Their scheme operates as follows:

Pre-distribution: The authority chooses a random symmetric, bivariate polynomial $F \in \mathbb{Z}_p[x, y]$ of degree t in each variable. It also chooses at random two univariate degree-t "noise polynomials" $g(y), h(y)$ over $\mathbb{Z}_p$. Let Small be the set of points for which both $g(y)$ and $h(y)$ are small; that is:

$$\mathsf{Small} \stackrel{\text{def}}{=} \{y \in \mathbb{Z}_p : \; g(y), h(y) \in [0, r]\}.$$

For any fixed $y \in \mathbb{Z}_p$, the probability (over choice of g, h) that $y \in \mathsf{Small}$ is r^2/p^2. The authority finds (in time $\mathcal{O}(N \cdot p^2/r^2)$) a set of N points $x_1, \ldots, x_N$ in Small. For each node i, the authority chooses a random bit b_i and gives node i the point x_i and the univariate polynomial

$$s_i(y) = F(x_i, y) + b_i \cdot g(y) + (1 - b_i) \cdot h(y) \,.$$

Namely, the noise polynomial is chosen as either $g(y)$ or $h(y)$, depending on the random bit b_i.

Key agreement: To compute a shared secret key, nodes i and j exchange their points x_i, x_j and then node i computes $s_i(x_j) \bmod p$ and node j computes $s_j(x_i) \bmod p$. Since

$$s_i(x_j), s_j(x_i) \in \{F(x_i, x_j), F(x_i, x_j) + 1, \ldots, F(x_i, x_j) + r\} \,,$$

the points computed by the two parties are close enough that they can be used to obtain a shared key by taking, e.g, the high-order bits of their respective results. (Zhang et al. describe an interactive protocol to handle wraparound, but this is irrelevant for the attacks we describe.)

Suggested parameters. The expected size of Small (over random choice of g, h) is r^2/p, so we require $r^2/p \geq N$. (Zhang et al. suggest a way to guarantee that the size of Small is at least r^2/p, but this still requires $r^2/p \geq N$.) The shared key has length roughly $\log(p/r)$, but p/r cannot be too large since the predistribution phase requires $\mathcal{O}(N \cdot p^2/r^2)$ work. If a larger key is desired, multiple instances of the scheme can be run in parallel and the derived pairwise keys concatenated.

In Table 1 we list the parameters suggested by Zhang et al. We note that with these parameters, each node must store a secret key of size $\approx 250\kappa$ bits in order to compute κ-bit shared keys. For the same amount of storage, the original scheme of Blundo et al. would give information-theoretic

modulus p	noise r	# of nodes N	degree t	storage per node (per key-bit)
$2^{32}-5$	2^{22}	2^{12}	76	246 bits
$2^{36}-5$	2^{24}	2^{12}	77	246 bits
$2^{40}-87$	2^{26}	2^{12}	77	234 bits
$2^{40}-87$	2^{28}	2^{16}	77	273 bits

Table 1: The suggested parameters for the key predistribution scheme of Zhang et al.

resilience to compromise of about 250 nodes. In contrast, Zhang et al. claim (computational) resilience $t = N$, an improvement of 1–2 orders of magnitude. As we will see, this claim is unfounded and the scheme can be broken by an attacker who compromises $t+3 \leq 80$ nodes. Thus the scheme of Zhang et al. is *less* resilient (as well as less efficient) than the original scheme of Blundo et al.

2.3 Warm-Up: A Simple Attack Using Error Correction

We begin by describing a relatively simple attack on the scheme as described above. We discuss two variants: a very efficient attack that requires corruption of $\approx 4t$ nodes, and an attack that runs in time $\mathcal{O}(r)$ but requires corruption of only $\approx 3t$ nodes. Both attacks rely on the error-correction algorithm of Ar, Lipton, Rubinfeld, and Sudan [1].

The first attack works as follows: Compromise $n = 4t+1$ nodes with points $x_1, x_2, \ldots, x_n$ to obtain the n polynomials $s_1(\cdot), \ldots, s_n(\cdot)$. Choose a point $x^* \in \mathbb{Z}_p$ belonging to any non-compromised node v^*, and compute $y_i = s_i(x^*)$ for $i = 1, \ldots, n$. By construction of the s_i we have

$$s_i(x^*) = F(x_i, x^*) + b_i \cdot g(x^*) + (1 - b_i) \cdot h(x^*) = f^*(x_i) + \text{noise}_{b_i},$$

where $f^*(\cdot) \overset{\text{def}}{=} F(\cdot, x^*)$, $\text{noise}_0 \overset{\text{def}}{=} h(x^*)$, and $\text{noise}_1 \overset{\text{def}}{=} g(x^*)$.

Define $f_b^*(x) \overset{\text{def}}{=} f^*(x) + \text{noise}_b$. Considering the set of pairs $\{(x_i, y_i) : i = 1, \ldots, n\}$, we have that for all i either $y_i = f_0^*(x_i)$ or $y_i = f_1^*(x_i)$. Hence, for at least one of $b = 0$ or $b = 1$ we have $y_i = f_b^*(x_i)$ for at least $2t+1$ values of i. Applying the error-correction algorithm of Ar et al. [1], we can recover at least one of the polynomials $f_0^*(\cdot)$ or $f_1^*(\cdot)$.

Once we have either of these polynomials, we can compute the shared key between the node v^* (that is associated with the point x^*) and any other node in the network: to get the shared key between v^* and another node v' associated with the point x', we compute $y = f_b^*(x') = f^*(x') + \text{noise}_b$. Since $\text{noise}_b \in [0, r]$, we see that y is close to $f^*(x') = F(x', x^*)$. Hence the high-order bits of y are (essentially) equal to the shared key between the two nodes.

A variant of the attack, which requires corrupting only $\approx 3t$ nodes, is as follows. Define x^*, f^*, f_b^*, and noise_b as above; here, set $n = 3t+1$. Assume without loss of generality that $\text{noise}_1 > \text{noise}_0$, and treat the value $\delta \overset{\text{def}}{=} \text{noise}_1 - \text{noise}_0$ as known. (Enumerating over all possible values of δ increases the running time by a multiplicative factor of r.)

Corrupt n nodes and compute the set $S = \{(x_i, y_i) : i = 1, \ldots, n\}$ as above. Construct S' by adding to S an additional n tuples $\{(x_{n+i}, y_{n+i})\}$ where $x_{n+i} = x_i$ and $y_{n+i} = y_i - \delta$. Observe that for every tuple $(x_i, y_i) \in S'$ it holds that either $f_0^*(x_i) = y_i$, or $f_1^*(x_i) = y_i$, or $f_0^*(x_i) - \delta = y_1$. Moreover, for $1 \leq i \leq n$, either $f_0^*(x_i) = y_i$ or else $f_0^*(x_{n+i}) = y_{n+i}$; thus, for exactly n tuples

$(x_i, y_i) \in S'$ it holds that $f_0^*(x_i) = y_i$. The error-correction algorithm of Ar et al. can thus be used to recover f_0^*. (The rest of the attack proceeds as before.)

For the parameters in Table 1 it always holds that $3t + 1 < 233$ and so this already shows that the scheme performs worse than the original, perfectly secure scheme of Blundo et al. In the following section we show that even a generalized version of the scheme that uses more noise (and is not susceptible to the attack described in this section) is vulnerable to attack, and moreover using only $\approx t$ corruptions.

2.4 A Generalized Scheme

The attack described in the previous section relies strongly on the fact that the same noise polynomial is used half the time. This suggests an easy patch that foils the attack described in the previous section: Let g, h, and Small be as in Section 2.2, and let u be "small" relative to p (we will see exactly how small below). Now for each node i choose random $\alpha_i, \beta_i \in [-u, u]$, and give to node i (with identity x_i) the univariate polynomial

$$s_i(y) = F(x_i, y) + \alpha_i \cdot g(y) + \beta_i \cdot h(y).$$

This generalizes the scheme of Zhang et al., since their scheme can be obtained by setting $\alpha_i = b_i$, $\beta_i = 1 - b_i$. The "error polynomial" $\alpha_i g(y) + \beta_i h(y)$ still evaluates to a value in a small range (namely, $[-2ur, 2ur]$) on every point in Small, and so this still allows every pair of parties to compute a shared key.

The noise is now larger than in the scheme of Zhang et al. by a factor of $4u$, so for the same values of p, r, t the pairwise keys will have roughly $\log(4u)$ fewer bits. Still one might hope that this modification would make the scheme more secure, even for small values of u. Unfortunately this is not the case, and below we present an attack that breaks also this more general scheme in time (roughly) $\mathcal{O}\big(t^3 + t \cdot (2u)^3\big)$ using only $t + 3$ compromised nodes. Note that $u = 1$ for the original scheme of Zhang et al., and so this gives a very efficient attack on their scheme (using fewer compromised nodes than the attack of the previous section). Furthermore, u cannot be too large: we need $4ur < p$ in order for even a single-bit shared key to be derived, meaning that in the worst case (for the attacker) the running time of the attack is $\mathcal{O}\big(t^3 + t \cdot (p/2r)^3\big)$. Unfortunately, the time required to initialize the scheme is $\mathcal{O}\big(N \cdot (p/r)^2\big)$, so p/r cannot be too large.

2.5 Outline of the Attack

In will be helpful in what follows to identify univariate polynomials of degree-t with vectors of length $t+1$. Specifically, we will identify the degree-t polynomial $p(y) = a_0 + a_1 y + \cdots + a_t y^t$ with its coefficient vector $\vec{p} = (a_0, a_1, \ldots, a_t)$.

Let $f_i(y) = F(x_i, y)$, where F is the bivariate polynomial chosen by the authority and x_i is the point associated with node i. The polynomial s_i given to node i is thus identified with the vector

$$\vec{s_i} = \vec{f_i} + \alpha_i \cdot \vec{g} + \beta_i \cdot \vec{h}\,.$$

The crucial observation underlying our attack is that the "noise" added to $\vec{f_i}$ is drawn from a low-dimensional linear subspace spanned by the two vectors $\vec{g}$ and $\vec{h}$. The attack proceeds by first identifying this "noise space", then finding the noise polynomials g and h, and finally solving for the bivariate polynomial F. We describe these steps in the three sections that follow.

2.6 Identifying the Noise Space

The attack begins by corrupting $n = t+3$ nodes with associated points $x_0, \ldots, x_{t+2}$. This gives a set of n vectors $\{\vec{s}_i\}_{i=0}^{t+2}$ with

$$\vec{s}_i = \vec{f}_i + \alpha_i \cdot \vec{g} + \beta_i \cdot \vec{h}\,.$$

The "noise space" is the vector space spanned by $\vec{g}$ and $\vec{h}$, and we now show how to identify this space. We use the fact that the $\vec{f}_i$ are all derived from the same bivariate polynomial F. Thus, if we write $F(x,y)$ as $F(x,y) = \sum_{j=0}^{t} F_j(x) \cdot y^j$ (where each F_j is a univariate degree-t polynomial), then for every node i we have

$$\vec{f}_i = (F_0(x_i), \ldots, F_t(x_i))\,.$$

Recall now the Lagrange interpolation formula: If P is a degree-t polynomial, then for any set of $t+1$ points $X = \{x_0, x_1, \ldots, x_t\}$ and any point x, it holds that

$$P(x) = \sum_{i=0}^{t} P(x_i) \cdot \underbrace{\prod_{j \neq i} \frac{x - x_j}{x_i - x_j}}_{L(X,x,i)}\,.$$

In particular, this formula applies to each of the polynomials F_j. If we compromise $t+3$ nodes with points $x_0, x_1, \ldots, x_t, x_{t+1}, x_{t+2}$ and set $X = \{x_0, x_1, \ldots, x_t\}$, then we have

$$\vec{f}_{t+1} - \sum_{i=0}^{t} L(X, x_{t+1}, i) \cdot \vec{f}_i = 0 \quad \text{and} \quad \vec{f}_{t+2} - \sum_{i=0}^{t} L(X, x_{t+2}, i) \cdot \vec{f}_i = 0. \tag{1}$$

Note that we can compute explicitly all the coefficients $L(X, x, i)$ in the equations above. Taking the same linear combinations of the $\{\vec{s}_i\}$ we get

$$\begin{aligned}
\vec{v} &\overset{\text{def}}{=} \vec{s}_{t+1} - \sum_{i=0}^{t} L(X, x_{t+1}, i) \cdot \vec{s}_i \\
&= \left(\alpha_{t+1} - \sum_{i=0}^{t} \alpha_i \cdot L(X, x_{t+1}, i)\right) \vec{g} + \left(\beta_{t+1} - \sum_{i=0}^{t} \beta_i \cdot L(X, x_{t+1}, i)\right) \vec{h} \\
&\in \operatorname{span}(\vec{g}, \vec{h})\,,
\end{aligned}$$

and similarly $\vec{v}\,' \overset{\text{def}}{=} \vec{s}_{t+2} - \sum_{i=0}^{t} L(X, x_{t+2}, i) \cdot \vec{s}_i \in \operatorname{span}(\vec{g}, \vec{h})$. Since the α's and the β's are chosen independently and uniformly from $[-u, u]$, it is easy to prove that $\vec{v}$ and $\vec{v}\,'$ span the entire space $\operatorname{span}(\vec{g}, \vec{h})$ except with probability at most $1/2u$. Experimentally, we find that $\vec{v}$ and $\vec{v}\,'$ span the entire space almost surely.

2.7 Finding g and h

Having computed two polynomials v, v' whose associated vectors $\vec{v}', \vec{v}\,'$ span the noise space, we now set out to find the original polynomials g and h. Here we use the fact that g, h are such that $g(x_i), h(x_i)$ are "small" (namely, in $[0, r]$) for all the x_i's.

Consider the n'-dimensional integer lattice Λ spanned by the rows of the following matrix:

$$\begin{bmatrix} v(x_0) & v(x_1) & \cdots & v(x_{n'-1}) \\ v'(x_0) & v'(x_1) & \cdots & v'(x_{n'-1}) \\ p & 0 & \cdots & 0 \\ 0 & p & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p \end{bmatrix}, \tag{2}$$

where $n' \leq t$ will be fixed later. Define $g^* \stackrel{\text{def}}{=} (g(x_0), \ldots, g(x_{n'-1}))$ and $h^* \stackrel{\text{def}}{=} (h(x_0), \ldots, h(x_{n'-1}))$ (where the polynomial evaluation is done modulo p), and note that these are both short vectors (of length at most $r \cdot \sqrt{n'}$) in this lattice.

We argue in Appendix A.1 (and verified experimentally) that when n' is large enough so that $p^2 \cdot (4r/p)^{n'} < 1$, then with high probability the two shortest (independent and non-zero) vectors in the lattice Λ are $\pm(g^* - h^*)$ and the smaller of g^* or h^*. This allows us to recover g^*, h^* (and hence the polynomials g and h) using lattice-basis reduction, as described next. Observe that to ensure $p^2 \cdot (4r/p)^{n'} < 1$, it is sufficient to set $n' > \left\lceil \frac{2 \log p}{\log p - \log 4r} \right\rceil$, which is independent of the degree t. For the parameters suggested in [7] using $n' = 11$ is always enough. For this small dimension, standard lattice-reduction algorithms can exactly compute all the small vectors in the lattice, including the two shortest vectors that we need.

Denote by ℓ_1, ℓ_2 the two shortest (independent and non-zero) vectors in Λ. As we said above, with high probability one of these vectors is $\pm(g^* - h^*)$ and the other is the shorter of g^* or h^*. In other words, with high probability the original vectors g^*, h^* belong to the set $\{\ell_1, \ell_2, \pm(\ell_1 \pm \ell_2)\}$. We can identify g^*, h^* using the fact that $g^*, h^* \in [0, r]^{n'}$. (In fact, g^*, h^* are uniform in $[0, r]^{n'}$ since the polynomials g, h are random and the x_i's are chosen subject to the constraint $g(x_i), h(x_i) \in [0, r]$. Thus, with high probability the only vectors in the set $\{\ell_1, \ell_2, \pm(\ell_1 \pm \ell_2)\}$ that belong to $[0, r]^{n'}$ are the original g^* and h^*.) So, given ℓ_1, ℓ_2 the original vectors g^*, h^* can be easily found.

2.8 Solving for F

Once we have recovered g and h, we can solve for F itself. Recall that each of the s_i obtained from a compromised node satisfies

$$\vec{s_i} = \vec{f_i} + \alpha_i \cdot \vec{g} + \beta_i \cdot \vec{h},$$

where $\alpha_i, \beta_i \in [-u, u]$. Using the fact that F is symmetric, we have

$$s_i(x_j) - \alpha_i \cdot g(x_j) - \beta_i \cdot h(x_j) \;=\; f_i(x_j) = f_j(x_i) \;=\; s_j(x_i) - \alpha_j \cdot g(x_i) - \beta_j \cdot h(x_i) \tag{3}$$

for all $i \neq j$. Having compromised $n = t + 3$ nodes, this gives a set of $\binom{n}{2}$ linear equations in the $2n$ unknowns $\{\alpha_i, \beta_i\}_{i=0}^{n-1}$. Naively, we would expect this system to have full rank when $\binom{n}{2} \geq 2n$, in which case we could solve for all the α_i, β_i and then recover the f_i and F itself. However, this is not the case: the system is under-defined, even if we add to the system the constraints from Eq. (1). In fact, the space of solutions to this system of equations turns out to have dimension exactly three, irrespective of t or n. (See Appendix A.2 for an explanation.)

Since we know that $\alpha_i, \beta_i \in [-u, u]$ for all i, we can exhaustively search for the desired solution as follows: Set the values of three of the α's and β's to values in $[-u, u]$; then solve the linear system

p	r	t	u	setup time (minutes)	attack time (minutes)	successes/attempts
$2^{32}-5$	2^{22}	76	2	60	10	7/7
$2^{36}-5$	2^{24}	77	2	1060	8	2/2

Table 2: Successful attacks on the scheme of Zhang et al. Timings reflect an implementation in Sage, running on an Intel® Xeon® CPU X7460 @ 2.66GHz with 128GB RAM.

for the rest of the α's and β's and check whether they also lie in the desired range. (Heuristically, we expect that will overwhelming probability there will be a *unique* solution to the system of linear equations that also satisfies $\forall i: \alpha_i, \beta_i \in [-u, u]$, and this is confirmed by our experiments.) This exhaustive search can be done in time $\mathcal{O}\big(t^3 + t \cdot (2u)^3\big)$ by solving the system parametrically (in time $\mathcal{O}\big(t^3\big)$) and then enumerating through $(2u)^3$ settings of the first three α's and β's until the desired solution is found.

For large u, one could also use lattice-reduction techniques to eliminate the exhaustive search for the α's and β's. This follows from the observation that the set of solutions to our linear system forms a dimension-three integer lattice, and the desired solution of α's and β's is a short vector in that lattice.

2.9 Experimental Verification

We implemented our attack both in C++ using NTL (`http://shoup.net/ntl`) and in Sage [4] using Damien Stehlé's fpLLL implementation (`http://perso.ens-lyon.fr/damien.stehle`) to carry out the LLL reduction. The source code of our attack (in Sage) is available on-line at `http://www.bitbucket.org/malb/algebraic_attacks/noise_poly.py`. Our attack ran quickly, and was successful the vast majority of the time; see Table 2 for representative results. Note that what prevented us from carrying out our attack on larger parameter sets was not the time required for the attack, but the time required to initialize the system!

2.10 Adding More Noise in The Free Term

A further generalization of the scheme of Zhang et al. would be to add more noise in the free term of the secret polynomials, setting

$$s_i(y) = F(x_i, y) + \alpha_i \cdot g(y) + \beta_i \cdot h(y) + \gamma_i,$$

where $\alpha_i, \beta_i \in_R [-u, u]$ and $\gamma_i \in_R [-ur, ur]$. Our attack can be easily adapted to break this variant:

- In the first stage, we recover the noise space but ignore the free term. That is, we recover the same two polynomials v, v', but instead of having $\vec{g} = a \cdot \vec{v} + b \cdot \vec{v}\,'$ for some scalars a, b, we would have $\vec{g} = a \cdot \vec{v} + b \cdot \vec{v}\,' + c \cdot \langle 1, 0, \ldots, 0\rangle$ for some a, b, c (and similarly for h).
- Instead of the lattice from Eq. (2), we use the lattice that is spanned by the rows of the

following matrix:

$$\begin{bmatrix} v(x_1)-v(x_0) & v(x_2)-v(x_1) & \cdots & v(x_{n'})-v(x_{n'-1}) \\ v'(x_1)-v'(x_0) & v'(x_2)-v'(x_1) & \cdots & v'(x_{n'})-v'(x_{n'-1}) \\ p & 0 & \cdots & 0 \\ 0 & p & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p \end{bmatrix}$$

Note that the values $v(x_i) - v(x_{i-1})$ are independent of the free term of v (and similarly for v'), and that the short vectors in this lattice correspond to the vectors

$$\tilde{g} = \langle g(x_0) - g(x_1), \ldots, g(x_{n'}) - g(x_{n'-1}) \rangle \text{ and } \tilde{h} = \langle h(x_0) - h(x_1), \ldots, h(x_{n'}) - h(x_{n'-1}) \rangle .$$

For this lattice, the two shortest vectors (that can be obtained using lattice reduction) are $\pm\tilde{g}$ and $\pm\tilde{h}$ themselves, which allow recovery of the polynomials $\pm g$ and $\pm h$ except for the free terms. The free terms can then be approximated by any scalars that force $g(x_i)$, $h(x_i) \in [0, r]$ for all the x_i.

- The system of equations for the coefficients of F now includes the additional unknowns γ_i, and the degree of the solution space would be six rather than three; see Appendix A.2. (It is also possible to eliminate the γ_i's from the system and arrive at a system that has only three degrees of freedom as before.) Solving this system would not determine the free term of F, but the free term of F can be approximated by any scalar that makes $F(x_i, x_j)$ close enough to $s_i(x_j)$ for all i, j.

3 The Message Authentication Schemes of Zhang et al.

Zhang, Subramanian, and Wang [6] proposed schemes for message authentication in sensor networks. They begin by describing an initial scheme, called Scheme-I in their paper, that allows a base station to authenticate a message for a set of nodes. This scheme is information-theoretically secure as long as a bounded number of messages are authenticated, and a bounded number of nodes are compromised. We describe this scheme here.

Let p be a prime. The master secret key, stored by the base station, is a bivariate polynomial $F \in \mathbb{Z}_p[x, y]$ of degree d_n in x and degree d_m in y. The secret key for a node i is the univariate polynomial $f_i(\cdot) \overset{\text{def}}{=} F(i, \cdot)$. The authentication tag for a message $m \in \mathbb{F}$ is the univariate polynomial $f'_m(\cdot) \overset{\text{def}}{=} F(\cdot, m)$. Node i can verify the tag f'_m on a message m by checking whether $f'_m(i) \overset{?}{=} f_i(m)$.

The master secret key can be recovered in its entirety if either $d_n + 1$ nodes are compromised, or if $d_m + 1$ messages are authenticated by the base station. If no nodes are compromised and at most d_m messages are authenticated, or if no messages have been authenticated and at most d_n nodes have been compromised, the scheme is information-theoretically secure (with probability of forgery $1/p$).

Zhang et al. present a series of extensions to this basic scheme in their paper. Scheme-II, as above, enables the base station to authenticate messages for the nodes (i.e., multicast), and Scheme-IV allows for authentication of messages between the nodes (i.e., many-to-many communication). Zhang et al. also propose a Scheme-III, but they themselves show that it is not secure.

3.1 Scheme-II and How to Break it

To enhance the security of Scheme-I, Zhang et al. suggest to add noise in the free term of the various polynomials. Specifically, fix a noise parameter $r < p/2$. The secret key of a node i is now the univariate polynomial $s_i(\cdot) = F(i, \cdot) + \gamma_i$, where γ_i is chosen uniformly in $[0, r]$. Similarly, the authentication tag for a message m is now the univariate polynomial $t_m(\cdot) = F(\cdot, m) + \gamma_m$, where γ_u is chosen uniformly in $[0, r]$. Node i verifies the authentication tag t_m on a message m by checking whether $|s_i(m) - t_m(i)| \leq r$, where elements of $\mathbb{Z}_p$ are viewed as being in the range $[-\lfloor p/2 \rfloor, \lfloor p/2 \rfloor]$. Zhang et al. claim that an attack on this scheme requires complexity at least $r^{\min\{d_m, d_n\}+1}$, even if an arbitrary number of nodes are compromised and an arbitrary number of messages are authenticated (cf. Theorems 3.2 and 3.3 in [6]).

This scheme is, in fact, easy to break. Noise is only introduced in the free term, so most of the coefficients of the master polynomial can be recovered by simple interpolation. If $F(x, y) = \sum_{i,j} F_i^j x^i y^j$, then by compromising d_n+1 nodes an attacker can recover all the F_i^j's with $j > 0$, and after seeing d_m+1 authentication tags an attacker can recover all the F_i^j's with $i > 0$. Compromising $d_n + 1$ nodes and seeing $d_m + 1$ authentication tags thus allows the attacker to recover all the coefficients of F except for the free term. The free term can then approximated by finding any element of $\mathbb{Z}_p$ for which the resulting polynomial $F(x, y)$ gives node keys and authentication tags whose free term is close to the free term of the keys and tags already observed.

3.2 Scheme-IV and How to Break It

Scheme-III and Scheme-IV in [6] were designed to authenticate many-to-many communication. These schemes extend Scheme-I by using a tri-variate master polynomial whose three variables correspond to senders, receivers, and messages. Namely, the master key of the underlying scheme is a polynomial $F(x, y, z)$. A node i is given two secret keys: the bivariate polynomial $F(i, \cdot, \cdot)$ (to be used when it acts as a sender), and the bivariate polynomial $F(\cdot, i, \cdot)$ (for when it acts as a receiver). The tag for a message m sent by node i is the univariate polynomial $F(i, \cdot, m)$; and a receiver j verifies this tag in the obvious way. Scheme-III is obtained from this underlying scheme by adding noise to the free term, but Zhang et al. observe that the resulting scheme is not secure. Hence, in Scheme-IV they adopt the perturbation polynomial technique from [7] as described next.

In Scheme-IV there are noise parameters u, r with $u < r < p/4$. The master secret is again a tri-variate polynomial $F \in \mathbb{Z}_p[x, y, z]$ of degree d_s (the "sender degree") in x, degree d_r (the "receiver degree") in y, and degree d_m (the "message degree") in z. Two univariate "noise polynomials" $g(x)$ (of degree d_s) and $h(y)$ (of degree d_r) are also chosen. These define the sender ID-space Small_S (resp., the receiver ID-space Small_R), which contains all the points on which the value of g (resp., h) is "small"; i.e.,

$$\mathsf{Small}_S \stackrel{\text{def}}{=} \{x : g(x) \in [0, r/u]\} \text{ and } \mathsf{Small}_R \stackrel{\text{def}}{=} \{y : h(y) \in [0, r/u]\}.$$

The scheme works as follows:

- Each node is given two keys: one for when it acts as a sender, and one for when it acts as a receiver.

  The sender secret key consists of an identity $i \in \mathsf{Small}_S$ and the bivariate polynomial $a_i(\cdot, \cdot) = F(i, y, z) + \alpha_i \cdot h(y) + \beta_i$, with α_i chosen uniformly in $[0, u]$ and β_i chosen uniformly in $[0, r/2]$.

Similarly, the receiver secret key consists of an identity $j \in \mathsf{Small}_R$ and the bivariate polynomial $b_j(x,z) = F(x,j,z) + \gamma_j \cdot g(x) + \delta_j$ with γ_j chosen uniformly in $[0,u]$ and δ_j chosen uniformly in $[0,r]$.

- The authentication tag computed by a node with sender-ID i on the message m consists of the identity i and the univariate polynomial

$$t_{i,m}(y) \overset{\text{def}}{=} a_i(y,m) + \eta_m = F(i,y,m) + \alpha_i \cdot h(y) + \beta_i + \eta_m,$$

where η_m is chosen uniformly in $[0, r/2]$. A node with receiver-ID j verifies this tag by checking that $|b_j(i,m) - t_{i,m}(j)| \le 2r$.

This scheme can be broken much as in the case of Scheme-II. A key observation is that noise is only introduced in the coefficients that are independent of the message-variable z. Partition the master polynomial into one polynomial that depends on z and another that does not:

$$F(x,y,z) = \sum_{i,j,k} F_{i,j,k} x^i y^j z^k \quad = \quad \underbrace{\sum_{k=1}^{d_z} z^k \sum_{i,j} F_{i,j,k} x^i y^j}_{F1(x,y,z)} + \underbrace{\sum_{i,j} F_{i,j,0} x^i y^j}_{F2(x,y)} \ .$$

Let $h(y) = \sum_j h_j \cdot y^j$. Then the secret key for node with sender-ID w is

$$a_w(y,z) = \sum_{j=0}^{d_y} \sum_{k=1}^{d_z} \left(\sum_{i=0}^{d_x} F_{i,j,k} w^i \right) y^j z^k + \sum_{j=1}^{d_y} \left(\alpha_w h_j + \sum_{i=0}^{d_x} F_{i,j,0} w^i \right) y^j + \left(\alpha_w h_0 + \beta_w + \sum_{i=0}^{d_x} F_{i,0,0} w^i \right).$$

Observe that for $k > 0$, the coefficient of $y^j z^k$ in a_w depends only on $F1$ and not on the noise. Similarly, for $k > 0$ the coefficient of $x^i z^k$ in the receiver polynomial $b_w(x,z)$ depends only on $F1$ and not on the noise. This means that once the attacker compromises $d_x + 1$ senders or $d_y + 1$ receivers, it can fully recover the polynomial $F1$. Then, the only part of the master secret key that the attacker is missing is $F2(x,y)$, which is independent of the message variable z. This allows easy forgery, as described next.

Given a tag $t_{x^*,m}(y)$ computed by a non-compromised sender x^* on a message m, the attacker (who knows $F1$) can compute the polynomial

$$\Delta(y) \overset{\text{def}}{=} t_{x^*,m}(y) - F1(x^*,y,m) \quad = \quad F2(x^*,y) + \alpha_{x^*} h(y) + \beta_{x^*} + \eta_m \, .$$

The attacker can now forge the tag of any message m' as sent by the same x^*, by setting

$$\tilde{t}_{x^*,m'}(y) \ \overset{\text{def}}{=} \ F1(x^*,y,m') + \Delta(y) \qquad \Big(= \ F1(x^*,y,m') + F2(x^*,y) + \alpha_{x^*} h(y) + \beta_{x^*} + \eta_{m^*} \Big).$$

Note that this is exactly the tag that the sender x^* would have sent if it chose $\eta_{m'} = \eta_{m^*}$, which means that this is a valid tag for m' and would therefore be accepted by all the receivers.

Alternatively, the attacker can apply an attack similar to the one from Section 2 (using the fact that g, h have "small values" on all the identities) to recover also the remaining master polynomial $F2(x,y)$, and thereafter it can forge messages for any sender. We omit the details.

4 The Storage/Retrieval Schemes of Subramanian et al.

Subramanian, Yang, and Zhang [5] presented three schemes for the management of encryption (and decryption) keys, which can in turn be used for protecting sensitive information stored in sensor nodes. Below we consider the third scheme from [5], which uses a variant of perturbation polynomials. That scheme is quite involved and contains many details that are not relevant to our attacks. Hence, we first present a simplified scheme and show how to attack it (cf. Section 4.1), and then explain why the same attacks apply to the full scheme of Subramanian et al.

Roughly speaking, Subramanian et al. assume a network where nodes are initialized before deployment, and then deployed to the field where they operate unattended, collecting information from their environment and then encrypting it and storing it locally. The nodes are willing to send their encrypted data to users who request it, but only users with the appropriate keys can decrypt the information.

The lifetime of the system is partitioned in a series of phases, and nodes update their keys from one phase to the next. The goal of the third scheme from [5] is to be able to provide a user with keys that can be used to decrypt the data from all the nodes in phase i but not any other phase, while minimizing the storage and communication requirements and maximizing the resilience to node compromise and/or user compromise.

Underlying the third scheme from [5] is the following polynomial-based solution: The master key is a bivariate polynomial $F \in \mathbb{F}[x, y]$. The secret key of node u is the polynomial $f_u(\cdot) = F(u, \cdot)$, and the encryption key used by node u in phase v is $K_{u,v} = f_u(v) = F(u, v)$. A user that needs the keys for phase v is given the polynomial $g_v(\cdot) = F(\cdot, v)$ that can be used to compute the keys for all the nodes at this phase as $g_v(u) = F(u, v) = K_{u,v}$.

The problem with this noise-free solution is resilience. Specifically, Subramanian et al. identified the following three attack scenarios:

- When a node u is compromised, the polynomial $f_u(\cdot)$ is recovered and the attacker can compute the key $K_{u,v}$ used by u in every phase v. Ideally, we would like the encryption at the nodes to be *forward secure* so that compromising node u at phase v will not allow the attacker to decrypt the storage from any prior phases.
- If F has degree d_x in x, then once the attacker compromise $d_u + 1$ nodes it can recover the entire master polynomial.
- Similarly, if F has degree d_y in y, then once a user is given g_v for $d_v + 1$ different phases it can recover the entire master polynomial.

4.1 A Simple Noisy Scheme and How to Break it

To overcome the problems mentioned above, Subramanian et al. proposed to add noise to the free terms of the relevant polynomials. The system is again defined over $\mathbb{F} = \mathbb{Z}_p$ for some prime p, and we have a noise parameter $r \ll p$. The master key is a polynomial $F(x, y)$ of degree d_x in x and degree d_y in y; the secret key for node u is $s_u(\cdot) = F(u, \cdot) + \alpha_u$; and the user secret key for phase v is $t_v(\cdot) = F(\cdot, v) + \beta_v$, where α_u, β_v are scalars that are chosen uniformly at random in $[0, r]$. The encryption key for node u in phase v is taken to be the high-order bits of $s_u(v)$, which are essentially equal to the high-order bits of $t_v(u)$. (The exact mechanism by which a key is derived are not important for our attack.) Subramanian et al. suggest to use $p \approx 2^{64}$, $r = 2^{16}$, and a master polynomial of degree 15 in each variable.

This simple scheme does not address the forward-secrecy concern, but it is supposed to provide better resilience than the noise-free scheme from the previous section. Unfortunately, this is not the case. Similar to the attack described in Section 3.1, compromising $d_x + 1$ nodes allows an attacker to reconstruct the coefficients of the master polynomial $F(x, y)$ corresponding to all the terms $x^i y^j$ with $j > 0$, and learning user keys for $d_y + 1$ different phases allows an attacker to reconstruct the coefficients of the master polynomial $F(x, y)$ corresponding to all the terms $x^i y^j$ with $i > 0$. We now exhibit two different attacks based on this:

Attack 1: A simple attack, similar to the one from Section 3.2, requires learning one user key and compromising $d_x + 1$ nodes. Partition the master polynomial $F(x, y)$ into the part that depends on the phase-variable y and the part that does not; i.e., write $F(x, y) = F1(x, y) + F2(x)$. As noted above, an attacker that compromises $d_x + 1$ nodes can fully recover $F1(x, y)$. Given a user key for one phase v (namely, the polynomial $t_v(\cdot) = F(\cdot, v) + \beta_v$), the attacker can then compute the univariate polynomial

$$\Delta(x) \stackrel{\text{def}}{=} t_v(x) - F1(x, v) \quad \Big(= F(x, v) + \beta_v - F1(x, v) \;=\; F2(x) + \beta_v \Big).$$

This allows the attacker to derive a user key for any other phase v' as

$$\tilde{t}_{v'}(x) = F1(x, v') + \Delta(x) \quad \Big(= F(x, v') + \beta_v \Big).$$

As in the attack from Section 3.2, we observe that $\tilde{t}_{v'}(x)$ is a valid user key for phase v' and can thus be used to compute the encryption keys of all the nodes in this phase.

Attack 2: In this attack the attacker compromises $n + 1$ nodes for some $n > d_x$, but does not need to learn any user keys. Denote the IDs of the compromised nodes by $u_0, u_1, \ldots, u_n$. As before, the attacker uses the polynomials $s_{u_i}(\cdot)$ to recover the bivariate polynomial $F1(x, y)$ where $F(x, y) = F1(x, y) + F2(x)$. Then, choosing some arbitrary value v^*, the attacker can compute for every u_i a value

$$y_i = s_{u_i}(v^*) - F1(u_i, v^*) = F2(u_i) + \alpha_{u_i}\,. \tag{4}$$

Next, the attacker can try to recover $F2$ using the fact that all the α_{u_i}'s are small (i.e., in $[0, r]$). Consider the lattice defined by integer linear combinations of the rows of the following matrix:

$$\Lambda = \begin{bmatrix} y_1 - y_0 & y_2 - y_1 & \cdots & y_n - y_{n-1} \\ u_1 - u_0 & u_2 - u_1 & \cdots & u_n - u_{n-1} \\ u_1^2 - u_0^2 & u_2^2 - u_1^2 & \cdots & u_n^2 - u_{n-1}^2 \\ \vdots & \vdots & \ddots & \vdots \\ u_1^{d_x} - u_0^{d_x} & u_2^{d_x} - u_1^{d_x} & \cdots & u_n^{d_x} - u_{n-1}^{d_x} \\ p & 0 & \cdots & 0 \\ 0 & p & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p \end{bmatrix}. \tag{5}$$

(We also let Λ refer to the lattice itself.) Note that if we write $F2 = \sum_{i=0}^{d_x} f_i \cdot x^i$ then there exist integers $k_1, \ldots, k_n$ such that

$$(-1, f_1, f_2, \ldots, f_{d_x}, k_1, k_2, \ldots, k_n) \cdot \Lambda = (\alpha_{u_1} - \alpha_{u_0},\ \alpha_{u_2} - \alpha_{u_1},\ \ldots,\ \alpha_{u_n} - \alpha_{u_{n-1}}),$$

which is a “short” vector in the lattice. The attacker can thus use lattice-reduction tools to find the coefficients of $F2$ (except for the free term). A heuristic argument similar to the one in Appendix A.1 suggests that when n is large enough so that $p^{(d_x+1)} \cdot (4r/p)^n < 1$ then the vector corresponding to the α's is indeed the shortest vector in this lattice. With the parameters suggested by [5] (i.e., $d_x = 15$, $p \approx 2^{64}$, and $r = 2^{16}$), we need $n > (d_x+1)\log(p)/(\log(p) - \log(4r)) = (16 \cdot 64)/(64 - 18) \approx 22$. We verified experimentally that the attack works for those parameters even for $n = 22$.

Once the attacker recovers the coefficients $f_1, \ldots, f_{d_x}$ of $F2$, it can approximate the free term of $F2$ as in Section 3.1. This gives it a good enough approximation to the master polynomial F.

4.2 The Scheme of Subramanian et al. and How to Break it

The actual scheme from [5] has many additional components on top of the simple scheme from the previous section, but these components have no real impact on the attacks that we have described. Below we list these additional components and show how the attacks can be tweaked to accommodate them.

Many copies of the scheme. As described in [5], each node stores several univariate node-polynomials, corresponding to several copies of the scheme. At any time, one scheme is “active” (i.e., used to encrypt data) while others are “dormant”. Every so often, the central key-distribution center broadcasts some message that causes all the nodes to activate the next copy of the scheme, and then erase all the keying material from the previously active copy. (Of course, user keys for the current phase always correspond to the appropriate phase of the currently active copy.)

This has very little effect on the attacks: an attacker that compromises some nodes learns the relevant information for all the copies that are stored on these nodes at the time of compromise. By compromising sufficiently many nodes, it learns the node-key material corresponding to the currently active scheme, as well as to all the schemes that are still dormant. The only thing that the attacker cannot do is attack copies of the scheme that were already erased. Also, to mount Attack 1 from the previous section against some copy of the scheme, the attacker must wait until that scheme becomes active (since that attack requires the attacker to learn one user-key that belongs to the copy under attack).

Forward secrecy between phases. As pointed out above, the simple scheme from Section 4.1 does not address forward secrecy. Subramanian et al. use the following simple trick to obtain forward secrecy: when a copy of the scheme is activated, the center announces the interval of phases (denoted $[v_s, v_e]$) to be used with that copy. Each node u then derives the keys for all these phases, setting $K_{u,v}$ as the high-order bits of $s_u(v)$ for all $v \in [v_s, v_e]$; then the node u erases the polynomial s_u corresponding to this copy of the scheme and stores only the keys $K_{u,v}$. Thereafter, the key $K_{u,v}$ is used during phase v and is erased when that phase is over.

The attacks from Section 4.1 can still be applied to all the dormant copies, since for these copies the nodes must still store the polynomial $s_u(\cdot)$. As for the active copy of the scheme, depending on the size of the interval $[v_s, v_e]$ it may be possible to recover the polynomial $s_u(\cdot)$ from the $K_{u,v}$'s by using techniques similar to Attack 2 above. Namely, after corrupting some node u we have $K_{u,v} = s_u(v) + \rho_v$ where s_u has degree d_y and the ρ_v's are all small (i.e., in $[0, r]$). As long as we have sufficiently many of these v's, we can use the same lattice reduction techniques as in Attack 2 to recover s_u.

Forward-secrecy within a phase. Instead of using the same key $K_{u,v}$ to encrypt all the information during phase v, Subramanian suggested using the hashed key $H^i(K_{u,v})$ to encrypt the i'th piece of information. The only effect of this on our attacks is that the key $K_{u,v}$ is erased earlier from the memory of node u.

Polynomial one-time pad. Another component of the scheme in [5] that is different from the simple scheme described earlier is that the nodes do not store the node-polynomial $s_u(\cdot)$ explicitly. Essentially, a node u stores for each copy of the scheme a "pad polynomial" $p_u(\cdot)$ derived from a "master pad polynomial" $p(u, v)$ by setting $p_u(\cdot) = p(u, \cdot) - \alpha_u$ for a random $\alpha_u \in [0, r]$. To activate the next copy of the scheme, the center chooses at random the master polynomial $F(x, y)$ for the scheme and broadcasts to all the nodes the bivariate polynomial $s(x, y) = p(x, y) + F(x, y)$; node u then computes its node-polynomial as $s_u(\cdot) = s(u, \cdot) - p_u(\cdot)$.

It is clear that this has no real effect on our attacks. Upon compromising a node, the attacker learns the pad polynomials used by that node. The attacker can apply Attack 2 directly, thus recovering the "master pad polynomial" $p(x, y)$. The attacker then waits until the center broadcasts $s(x, y)$ and recovers $F(x, y) = s(x, y) - p(x, y)$. Alternatively, the attacker can wait until that copy is activated, compute all the $s_u(\cdot)$'s just as the nodes do, and recover $F(x, y)$ directly using Attack 1 or Attack 2.

Miscellaneous. In the actual scheme that is described in [5], the handling of the pad polynomials is slightly obfuscated as follows:

- The "master pad polynomials" for the different copies of the scheme are not necessarily independent. Rather, they are all derived from the same tri-variate polynomial $p^*(x, y, z)$, where the master pad for the k'th copy of the scheme is set as $p(\cdot, \cdot) = p^*(\cdot, \cdot, k)$. We did not find in [5] a specification of the copy-degree of p^* (i.e., the degree of the copy variable z). Hence, we assume that it is taken as large as the number of copies, so that the copies are truly independent. (This has no effect on any of the efficiency parameters that are discussed in [5].) If the copy-degree is smaller, then it is likely that one can find more attacks, where compromising some copies of the scheme allows the attacker to break also other copies.

- The noise in the free term is added in several steps rather than all at once (and is not quite independent between the different copies of the scheme). Specifically, the pad polynomials are computed by first setting $\tilde{p}_u(y, z) = p^*(u, y, z) - \alpha'_u$, and then the pad for the k'th copy is set as $\bar{p}_{u,k}(y) = \tilde{p}_u(y, k) - \alpha''_{u,k}$ with $\alpha'_u, \alpha''_{u,k}$ all chosen uniformly in $[0, r/4]$.[1]

- When activating the k'th copy, the center chooses a random master polynomial $F_k(x, y)$ for this copy and broadcasts a noisy bivariate polynomial $s_k(x, y) = p^*(x, y, k) + F_k(x, y) + \gamma$, where γ is uniform in $[0, r/2]$. Each node u computes its node-polynomial for this copy as $s_{u,k}(\cdot) = s_k(u, \cdot) - \bar{p}_{u,k}(\cdot)$.

Obviously, these details have no bearing on our attacks.

[1]This means that the pad polynomials $\bar{p}_{u,k}(y)$ introduce somewhat smaller noise of up to only $r/2$ in the free term, which makes our Attack 2 from Section 4.1 marginally easier.

5 Conclusion

We have shown attacks on the schemes from [7, 5, 6], which are all based on "perturbation polynomials". Our attacks show that the modified schemes are no better — and may, in fact, be worse — than the information-theoretically secure schemes they are based on. Our results cast doubt on the viability of the "perturbation polynomials" technique as an approach for designing secure cryptographic schemes.

Note: The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

References

[1] S. Ar, R. Lipton, R. Rubinfeld, and M. Sudan. Reconstructing Algebraic Functions from Mixed Data. *SIAM J. Computing* 28(2): 487–510, 1998.

[2] R. Blom. An Optimal Class of Symmetric Key Generation Systems. Eurocrypt '84.

[3] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and Moti Yung. Perfectly Secure Key Distribution for Dynamic Conferences. *Information and Computation* 146(1): 1–23, 1998.

[4] W.A. Stein et al. *Sage Mathematics Software (Version 3.3)*. The Sage Development Team, 2009, `http://www.sagemath.org`.

[5] N.V. Subramanian, C. Yang, and W. Zhang. Securing Distributed Data Storage and Retrieval in Sensor Networks. 5th IEEE International Conference on Pervasive Computing and Communications (PerCom'07), 2007.

[6] W. Zhang, N. Subramanian, and G. Wang. Lightweight and Compromise-Resilient Message Authentication in Sensor Networks. IEEE INFOCOM, 2008.

[7] W. Zhang, M. Tran, S. Zhu, and G. Cao. A Random Perturbation-based Scheme for Pairwise Key Establishment in Sensor Networks. 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'07), 2007.

A Technical Details

A.1 The Shortest Vectors in the Lattice from Eq. (2)

Here we justify the claims made in Section 2.7. Recall the setting: we have an integer lattice Λ defined by taking integer linear combinations of the rows of the following matrix:

$$\begin{bmatrix} g(x_0) & g(x_1) & \cdots & g(x_{n'-1}) \\ h(x_0) & h(x_1) & \cdots & h(x_{n'-1}) \\ p & 0 & \cdots & 0 \\ 0 & p & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p \end{bmatrix}. \tag{6}$$

(We have substituted g, h in place of v, v'; since v, v' and g, h span the same space, the lattice is unchanged.) Recall that g^* (resp., h^*) denotes the first (resp., second) row of the lattice above, and that the length of g^*, h^* is at most $r \cdot \sqrt{n'}$. Let ℓ_1, ℓ_2 denote the two shortest (independent and non-zero) vectors in Λ. We provide a heuristic argument that $\ell_1, \ell_2 \in \{g^*, h^*, \pm(g^* - h^*)\}$ with high probability over the initial choice of g^*, h^*.

The polynomials g and h are chosen during system set-up as random polynomials of degree t, and the x_i are chosen such that $g(x_i), h(x_i) \in [0, r]$. Since $n' \le t$, the values of $g(x_0), \ldots, g(x_{n'-1})$ are independent and uniform in $[0, r]$ (and similarly for h). That is, g^* and h^* are independent and uniform in $[0, r]^{n'}$.

We note that we expect the vector $g^* - h^*$ to be shorter than both g^*, h^* (since the expected size of each entry in $g^* - h^*$ is roughly $r/3$, as compared to $r/2$ for each entry in g^*, h^*). We ask what is the probability that there exist some $a, b \in \mathbb{Z}_p$ (with $(a, b) \notin \{(0, \pm1), (\pm1, 0), \pm(1, -1)\}$) such that the vector $ag^* + bh^* \bmod p$ is shorter than both g^* and h^* (where the mod p operation maps integers into the range $[-\lfloor p/2 \rfloor, \lfloor p/2 \rfloor]$). We distinguish between "small pairs" where $|a|, |b| < p/4r$ and "large pairs" where at least one of $|a|, |b|$ is at least $p/4r$.

- For a "small pair", we have no reduction mod p (since $ag^* + bh^*$ is already in the range $[-p/2, p/2]$). Hence, there exists a "small pair" as needed if and only if the integer lattice that is spanned by only two vectors g^*, h^* contains a vector other than $\pm(g^* - h^*)$ which is shorter than both g^*, h^*. When $(a, b) \notin \{(0, \pm1), (\pm1, 0), \pm(1, -1)\}$, the expected size of each entry in $ag^* + bh^*$ is larger than $r/2$ and so we expect $ag^* + bh^*$ to be longer than g^*, h^*. Hence, we expect the two shortest vectors in the lattice spanned by g^*, h^* to be in the set $\{\pm g^*, \pm h^*, \pm(g^* - h^*)\}$ except with probability exponentially small in n'.

- For any fixed "large pair" (a, b), the distribution of the vector $ag^* + bh^* \bmod p$ is rather close to the uniform distribution over $[-\lfloor p/2 \rfloor, \lfloor p/2 \rfloor]$. To see this, assume $|a| > p/4r$. Fix b and h^* to some arbitrary values, and consider the residual distribution on $ag^* + bh^* \bmod p$ induced by choosing $g^* \in [0, r]^{n'}$. Consider a "simplified setting" where the entries of g^* are chosen from the real interval $[0, r]$ (instead of only the integers in this interval). In this setting, each entry of $ag^* + bh^* \bmod p$ would be chosen from a distribution that has statistical distance at most $3/4$ from the uniform distibution on $[-p/2, p/2]$. (When $|a|$ is larger still, the distibution gets even closer to uniform. For example, for $|a| = \Theta(p)$ the distance from uniform is $O(1/r)$.) The quantization to integers of course changes the distribution, but does not change substantially the probability that the resulting vector is short.

  Making the heuristic assumption that the length of $ag^* + bh^* \bmod p$ is distributed as if that vector were uniform in $[-\lfloor p/2 \rfloor, \lfloor p/2 \rfloor]^{n'}$, we can estimate the probability that this vector lies in the ball of radius $r\sqrt{n'}$ around the origin. The volume of such a ball is

$$\frac{(r\sqrt{n'})^{n'} \pi^{n'/2}}{(n'/2)!} \approx r^{n'} (2\pi e)^{n'/2} \approx (4r)^{n'}.$$

Hence, the probability that a uniformly distributed vector in $[-p/2, p/2]^{n'}$ has length below $r\sqrt{n'}$ is upper-bounded by $(4r/p)^{n'}$, and we can heuristically use the same bound also for the length of $ag^* + bh^* \bmod p$ for any fixed "large pair" (a, b). As there are fewer than p^2 "large pairs", a union bound implies that when $p^2 \cdot (4r/p)^{n'} \ll 1$ we expect to have $|ag^* + bh^* \bmod p\,| \geq r\sqrt{n'}$ for every "large pair".

Experimentally, we observe that for even moderate values of n', the two smallest vectors in the lattice are indeed $\pm(g^* - h^*)$ and the smaller of g^*, h^*. Specifically, we ran the following experiment: generate g^*, h^* uniformly in $[0, r]^{n'}$ and then run LLL on the lattice from Eq. (6) to compute the shortest vectors ℓ_1, ℓ_2 of the resulting lattice. Call it a "success" if $g^*, h^* \in \{\pm\ell_1, \pm\ell_2, \pm(\ell_1 \pm \ell_2)\}$. For each setting of p and r, we then determined the minimum value of n' for which a success occurred at least 95% of the time (in 200 trials). The results are in Table 3.

p	r	n'	p	r	n'
	2^{21}	9		2^{26}	9
	2^{22}	9		2^{27}	9
$2^{32} - 5$	2^{23}	10	$2^{40} - 87$	2^{28}	9
	2^{24}	10		2^{29}	10
	2^{25}	11		2^{30}	10

Table 3: Dimension n' needed for recovery of g^*, h^*.

A.2 Solving for the α's and β's

Here we explain why the linear system of equations described in Section 2.8 is under-defined, and why the vector space of solutions has dimension 3.

Every solution to our system of linear equations must correspond to a bivariate degree-t polynomial F (due the the inclusion of Eq. (1)) which is symmetric (due to the equations from Eq. (3)). Moreover, for each node associated with the point x_i the polynomial F induces coefficients $\vec{f_i}$ such that $\vec{s_i} - \vec{f_i}$ belongs to the vector space spanned by $\vec{g}$ and $\vec{h}$. Let F, F' be two polynomials satisfying these constraints, and consider their difference polynomial $D = F - F'$. This polynomial D satisfies the following three conditions:

- D is a bivariate degree-t polynomial (since F and F' are);
- D is symmetric (since F and F' are);
- For every i, if we let $\vec{d_i}$ denote the coefficients of the univariate polynomial $D(x_i, \cdot)$, then all the $\vec{d_i}$'s belong to the vector space spanned by $\vec{g}$ and $\vec{h}$.

We now show that there are exactly three degrees of freedom in choosing a polynomial D with these properties. Denote the matrix of coefficients of D by $[D]$, and denote by $[d]$ the matrix whose

ith row is the vector $\vec{d_i}$ for $i = 0, 1, \ldots, t$. Then $[d] = V \cdot [D]$, where V is a Vandermonde matrix:

$$\underbrace{\begin{pmatrix} d_{0,0} & d_{0,1} & \ldots & d_{0,t} \\ d_{1,0} & d_{1,1} & \ldots & d_{1,t} \\ \vdots & \vdots & \ddots & \vdots \\ d_{t,0} & d_{t,1} & \ldots & d_{t,t} \end{pmatrix}}_{[d]} = \underbrace{\begin{pmatrix} 1 & x_0 & \ldots & x_0^t \\ 1 & x_1 & \ldots & x_1^t \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_t & \ldots & x_t^t \end{pmatrix}}_{V} \cdot \underbrace{\begin{pmatrix} D_{0,0} & D_{0,1} & \ldots & D_{0,t} \\ D_{1,0} & D_{1,1} & \ldots & D_{1,t} \\ \vdots & \vdots & \ddots & \vdots \\ D_{t,0} & D_{t,1} & \ldots & D_{t,t} \end{pmatrix}}_{[D]}$$

The conditions on the polynomial D translate to the conditions that $[D]$ is a $(t+1) \times (t+1)$ symmetric matrix, and that the rows of $[d]$ are in the vector space spanned by $\vec{g}$ and $\vec{h}$. The last condition can be expressed in matrix notation by saying that there exists a $(t+1) \times 2$ matrix X such that

$$[d] = X \cdot \begin{pmatrix} \vec{g} \\ \vec{h} \end{pmatrix}.$$

To obtain a D satisfying these conditions, choose an arbitrary symmetric 2×2 matrix R and set

$$[D] := \left(\vec{g}^T \mid \vec{h}^T\right) \cdot R \cdot \begin{pmatrix} \vec{g} \\ \vec{h} \end{pmatrix},$$

where $\vec{g}^T$ and $\vec{h}^T$ (the transpose of $\vec{g}$ and $\vec{h}$, respectively) are column vectors. This ensures that $[D]$ is a $(t+1) \times (t+1)$ symmetric matrix, and moreover

$$[d] = V \cdot [D] = \underbrace{V \cdot \left(\vec{g}^T \mid \vec{h}^T\right) \cdot R}_{X} \cdot \begin{pmatrix} \vec{g} \\ \vec{h} \end{pmatrix}$$

as needed. Since there are three degrees of freedom in choosing a symmetric 2×2 symmetric matrix, we get exactly three degrees of freedom for D.

We observe that if we had the additional noise in the free term (as in Section 2.10), then the noise space would be spanned by the three vectors $\vec{g}$, $\vec{h}$, and $\vec{e_1} = \langle 1, 0, \ldots, 0 \rangle$. In this case, we can use the exact same argument, except that the matrix R is a symmetric 3×3 matrix and so we have six degrees of freedom in choosing it. (However, the lower-right $t \times t$ sub-matrix of D is still rank-2, so once we eliminate the dependence on the free terms we can get back to a system with only three degrees of freedom.)