Journées PaRISTIC, LaBRI, 21-23 novembre 2005

ACI Sécurité et informatique

OCAM

Opérateurs Cryptographiques et Arithmétique Matérielle

Participants :
Codes (INRIA), Arénaire (LIP, INRIA),
Arithmétique Informatique (LIRMM)

Présentation : Nicolas Sendrier (INRIA Rocquencourt, projet CODES)

OCAM - Objectifs

- Mise en œuvre FPGA d'un algorithme de signature numérique
- Implantation matérielle des corps finis
- Implantation matérielle des cryptosystèmes basés sur les codes

Signature basée sur les codes

[CFS01] N. Courtois, M. Finiasz, et N. Sendrier, *How to achieve a McEliece-based Digital Signature Scheme*, Asiacrypt 2001

Avantages :
- signature de taille très faible (80 à 150 bits)
- vérification simple et rapide (1 μs à 20 ms)
- réduction de sécurité à des problèmes difficiles de codage

Inconvénients :
- production d'une signature très lente (10 s, écart type important)
- clé publique de grande taille (9 Mbits)

Signatures courtes

Les signatures courtes sont utiles
- sur des supports contraints ou de petite taille
- en cas de faible bande passante

Systèmes existants :
- CFS : 80 bits
- Quartz : 128 bits
- Couplages : 160 bits
- DSA, ECDSA : 320 bits
- SFLASH^{v3} : 469 bits
- RSA : 1024 bits

Signatures courtes – Exemples d'application

- numéros de série de logiciels
- autorisations pour cartes de crédit
- billets de transport électroniques
- timbres postes électroniques
- billets de banque, chèques
- documents d'identité
- ...

Cryptographie et codes de Goppa

Secret :

- Polynôme unitaire $g(z) \in \mathbf{F}_{2^m}[z]$ degré t
- Support $L = (\alpha_1, \ldots, \alpha_n)$ éléments distincts de $\mathbf{F}_{2^m}$
- Code de Goppa $\Gamma(L, g)$ (longueur $n = 2^m$, codimension $r = tm$)

Public :

- matrice de parité H de Γ (binaire $r \times n$)

Chiffrement de Niederreiter

Syndrome, fonction à sens unique, S : $\begin{array}{ccc} \mathbf{F}_2^n & \rightarrow & \mathbf{F}_2^r \\ y & \mapsto & yH^T \end{array}$

Opération inverse difficile : pour $s \in \mathbf{F}_2^r$, trouver y de poids minimal dans $\mathbf{F}_2^n$ tel que $S(y) = yH^T = s$.

Chiffrement Niederreiter, E : $\begin{array}{ccc} W_{n,t} & \rightarrow & \mathbf{F}_2^r \\ e & \mapsto & eH^T \end{array}$

La fonction $E()$ est injective avec $W_{n,t} \subset \mathbf{F}_2^n$ l'ensemble des mots de poids t

La fonction $E()$ est inversible avec l'aide des secrets $g(z)$ et L.

Signature basée sur les codes

Pour une signature, il faut prouver à partir de s quelconque dans $\mathbf{F}_2^r$ que l'on connait les secrets et que l'on peut inverser $E()$. Deux possibilités :

1. Générer à l'aide d'une fonction « aléatoire » $f()$, une suite $s_i = f(s,i)$ d'éléments de $\mathbf{F}_2^r$

   Signature : e de poids t, indice i tels que $eH^T = f(s,i)$

2. Trouver un mot e de poids minimal ($> t$ en pratique) tel que $eH^t = s$.

   Signature : e de poids $t + \delta$ (en pratique $\delta = 2$)

Nous considérons des instances où $n = 2^m = 2^{16}$, $t = 9$ et $r = 144$

Coût du calcul de la signature

Dans les 2 cas tout se passe comme on générait des instances indépendantes du problème d'inversion de $E()$ (il s'agit d'un décodage de t erreurs). Le calcul s'arrête à la première réussite (épreuve de Bernouilli).

$\longrightarrow$ moyenne et écart type du nombre d'essais valent $t! = 9! = 362880$

L'opération d'inversion de $E()$ est relativement simple et doit être répétée un grand nombre de fois.

$\longrightarrow$ intérêt d'une implantation matérielle.

Choix algorithmiques (1/2)

Dans le papier [CFS01] le choix s'était porté sur le premier mode de fonctionnement : générer des instances $s_i = f(s, i)$ à l'aide d'une fonction aléatoire $f()$.

Ce choix avait été fait car si $f()$ est une fonction aléatoire alors l'algorithme produira une signature avec certitude (modèle de l'oracle aléatoire).

Dans le second cas il faut que le syndrome initial s ne corresponde pas dans un coset dont le leader serait de poids ≥ 12 (probabilité 2^{-155}), auquel cas l'algorithme échoue après quelques heures de calcul.

Choix algorithmiques (2/2)

L'implantation matérielle du premier choix (génération des s_i) pose plusieurs problèmes

- Choisir et implanter la fonction $f()$ est délicat (le modèle de l'oracle aléatoire n'a pas de sens dans la « vraie vie »).
- la signature (couple (e, i)) est de longueur variable non bornée supérieurement. En pratique on va borner l'indice i, mais on a une probabilité d'echec non nulle.

En revanche avec la seconde solution (décodage complet)

- les instances de décodage sont faciles à générer,
- la signature produite est de longueur fixe.

Solution algorithmique retenue

Entrée : $s \in \mathbf{F}_2^r$
Pour i de 2 à n
    Pour j de 1 à $i-1$
        $s' \leftarrow s + c_i + c_j$ // c_i *la* i*-ème colonne de* H
        $e \leftarrow E^{-1}(s')$
        Si ($e \neq$ ECHEC) // e *un mot de poids* $t = 9$
            retourner $e + u^{(i)} + u^{(j)}$ // $9 + 2 = 11$ *positions* $\neq 0$
retourner « ECHEC »

($u^{(i)} \in \mathbf{F}_2^n$ est le mot avec un seul 1 en position i)

$E^{-1}(s')$ utilise le secret et échoue ou retourne un mot e de poids t tel que $s' = eH^T$. Le mot y retourné par l'algorithme vérifie $yH^T = s$.

La signature y est un mot de longueur $n = 2^{16}$ et de poids 11, qui s'écrit a priori à l'aide de $\log_2 \binom{2^{16}}{11} = 150.75 \approx 151$ bits.

Raccourcissement de la signature (1/3)

Au lieu de fournir y de poids 11 pour la signature, nous fournissons le mot e' contenant les 8 positions les plus « à droite » de y.

Nous aurons $e'H^T = s + c_i + c_j + c_l$ pour un certain triplet de positions (i, j, l) (les indices i et j sont ceux des boucles de l'algorithme de signature). L'algorithme suivant

Entrée : $s \in \mathbf{F}_2^r$, $e' \in \mathbf{F}_2^n$ de poids 8
$s' \leftarrow s + e'H^T$ // $s' = (c_i + c_j + c_l)H^T$
Pour i de 2 à n
    Pour j de 1 à $i - 1$
        Si $(s' + c_i + c_j \in H)$ // *i.e. est une colonne de* H
            retourner « signature correcte »
retourner « signature incorrecte »

vérifie une signature en un nombre d'itérations identique à celui nécessaire pour signer ($t! = 362880$ en moyenne).

Raccourcissement de la signature (2/3)

On découpe le support du code en 2^{12} zones contenant 16 positions chacune. Au lieu de 8 positions, la signature indiquera 8 zones contenant ces positions.

Soit H' la matrice de parité du code poinçonné en ces $8 \times 16 = 96$ positions. Nous avons (sans perte de généralité les positions effacées sont à droite) pour un certain U inversible :

$$UH = \begin{array}{|c|c|} \hline H'' & \begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix} \\ \hline H' & 0 \\ \hline \end{array} \qquad Us^T = \begin{array}{|c|} \hline \times \\ \vdots \\ \times \\ \hline s'^T \\ \hline \end{array}$$

On cherche à présent 3 colonnes c'_i, c'_j, c'_l de H' dont la somme vaut s'. Les indices i et j sont ceux des boucles de l'algorithme de signature.

Raccourcissement de la signature (3/3)

Si on pose $r' = 48$ la codimension du code poinçonné, s' est le syndrome modifié (de longueur 48). L'algorithme suivant

Entrée : $s' \in \mathbf{F}_2^r$
Pour i de 2 à $n - 96$
    Pour j de 1 à $i - 1$
        Si $(s' + c'_i + c'_j \in H')$ // *i.e. est une colonne de* H'
            retourner « signature correcte »
retourner « signature incorrecte »

vérifie une signature en un nombre d'itérations identique à celui nécessaire pour signer ($t! = 362880$ en moyenne).

Il existe $\binom{2^{12}}{8} = 2^{80.69}$ mots de longueur 2^{12} et de poids 8, donc la longueur de la signature raccourcie est de 81 bits.

Enfin, comme ces mots de poids 8 ne sont pas équirépartis, on peut coder la signature sur 80 bits avec une probabilité de 94% (donc un surcoût de 6% pour la signature).

Mise en œuvre software

Temps mesurés en software sur un Pentium 4, 3.2Ghz

- temps moyen de signature : 6.9 s, mini 0.22 s, maxi 83.3 s
- temps moyen de vérification : 21 ms, mini 4 ms, maxi 160 ms (vérification d'une fausse signature 2 s)

Ces temps sont problématiques pour la signature car la plupart des algorithmes utilisés signent et vérifient en quelques dizaines de millisecondes.

Mise en œuvre FPGA (1/3)

Circuit choisi : Xilinx XCV300E-7 (qq dizaines de dollars) ou équivalent

En pratique la clé publique est choisie sous forme systématique $H = (Id|R)$, le syndrome fourni au FPGA n'est pas $s = (s_1, \ldots, s_r) \in \mathbf{F}_2^r$ mais le syndrome modifié (double)

$$S_0(z) = \sum_{i=1}^{r} s_i C_{\alpha_i}(z), \qquad C_\beta(z) = \sum_{l=0}^{2t-1} \frac{\beta^l z^l}{g(\beta)^2} \in \mathbf{F}_{2^m}[z]$$

Mise en œuvre FPGA (2/3)

Initialisation : $(\gamma_i, 1/g(\gamma_i)^2)$, $1 \leq i \leq \text{MAX}$
Entrée : $S_0(z) \in \mathbf{F}_{2^m}[z]$ de degré $2t-1$
Pour i de 2 à MAX
    Pour j de 1 à $i-1$
        $S(z) \leftarrow S_0(z) + C_{\gamma_i}(z) + C_{\gamma_j}(z)$ // *modification du syndrome*
        $\sigma(z) \leftarrow$ Solution $\left(S(z) = \frac{\omega(z)}{\sigma(z)} \mod z^{2t}\right)$ // *Berlekamp-Massey*
        Si $\left(z^{2^m} - z = 0 \mod \sigma(z)\right)$ // *test de divisibilité*
            retourner (i, j)

À l'initialisation du circuit, on lui fournit MAX éléments du support (et les $1/g(\gamma_i)^2$ correspondants). La valeur de MAX est calculée pour que la probabilité d'echec soit négligeable. Pour MAX $= 6000$ cette probabilité est inférieure à 2^{-71}.

Mise en œuvre FPGA (3/3)

L'essentiel de la surface est utilisée par les multiplieur-additionneurs dans $\mathbf{F}_{2^{16}}$ (plus de 3% de la surface disponible pour chacun). L'arithmétique du corps fini utilise la structure récursive $\mathbf{F}_{2^{16}} = \mathbf{F}_{2^{2^{2^2}}}$.

Pour optimiser le chemin critique et le pipe-line, le calcul est divisé en trois partie, chacune d'elle utilise

- Modification du syndrome : 1 multiplieur-additionneur
- Berlekamp-Massey : 10 multiplieur-additionneurs + un inverseur
- Test de divisibilité : 9 multiplieur-additionneurs

Le reste du circuit est principalement dédié aux registres et au contrôle.

Le prototype est toujours en cours de réalisation, mais les premiers tests permettent d'espérer une signature en moins d'une seconde.

Conclusions

- Nous avons établi la faisabilité de la signature sur FPGA en moins d'une seconde
- Le problème algorithmique bloquant est l'arithmétique sur le corps fini
- Le changement de paramètres, de corps fini en particulier, demandera un effort supplémentaire conséquent
- L'optimisation de l'implantation matérielle a conduit à des choix algorithmiques qui n'étaient pas évidents initialement