Fast Multivariate Signature Generation in Hardware: The Case of Rainbow

Sundar Balasubramanian, Harold W Carter
Department of ECE
University of Cincinnati, Cincinnati, OH
balasusu@email.uc.edu, hal.carter@uc.edu

Andrey Bogdanov, Andy Rupp
Horst-Görtz Institute for IT-Security
Ruhr-University Bochum, Germany
{abogdanov, arupp}@crypto.rub.de

Jintai Ding
Department of Mathematical Sciences
University of Cincinnati, Cincinnati, OH
jintai.ding@uc.edu

Abstract

This paper presents a time-area efficient hardware architecture for the multivariate signature scheme Rainbow. As a part of this architecture, a high-performance hardware optimized variant of the well-known Gaussian elimination over GF(2^l) and its efficient implementation are presented. The resulting signature generation core of Rainbow requires 63,593 gate equivalents and signs a message in just 804 clock cycles at 67 MHz using AMI 0.35μm CMOS technology. Thus, Rainbow provides significant performance improvements compared to RSA and ECDSA.

1 Introduction

Multivariate Public-Key Cryptosystems (MPKCs) are cryptosystems for which the public key is a set of polynomials $P(X) = (p_1, \ldots, p_m)$ in variables $X = (x_1, \ldots, x_n)$ where all variables and coefficients are in a finite field GF(q). The security assumption of these cryptosystems relies on the NP-completeness of solving a set of multivariable quadratic **MQ** polynomial equations. Unlike the traditional cryptosystems such as the RSA and ECC, multivariate schemes are not yet shown vulnerable to the future quantum computer attacks.

Rainbow signature schemes [1] represent a family of multivariate signature schemes, which have great potential in terms of its efficiency and its application in ubiquitous computing devices. These schemes are based on the Unbalanced Oil-and-Vinegar multivariate structure [2], and the Rainbow class repeatedly applies of the Unbalanced Oil-and-Vinegar principle. Though there are a number of known attacks on the Rainbow family, they are not fundamental in the sense that all of them can be easily prevented by slightly adjusting the parameters suggested.

The paper at hand presents the first special-purpose hardware implementation of Rainbow. As a part of this core, we have also designed and implemented G-SMITH, a parallel hardware architecture for the specific case of Gaussian elimination over GF(2^l), which is a generalization of the SMITH architecture [3]. We propose an area-time efficient VLSI architecture for Rainbow signature generation. The time-area product of our implementation of Rainbow is 25 times better than that of en-TTS [4], making our architecture the most efficient one for a multivariate signature scheme. We also compared our system with the well-known RSA and ECDSA implementations and Rainbow's metric is superior to these schemes. Hence our experiments indicate that there is a lot of potential for deploying MQ schemes in practical applications. Our architecture consumes a total logic space of 63,593 GE and signs a message in just 804 cycles at 67 MHz using AMI 0.35μm CMOS technology.

2 The Rainbow Signature Scheme

The key mathematical technique used by Rainbow is a multi-layer oil-vinegar system, which is a quadratic system of equations involving two types of variables (oil and vinegar), where there are no quadratic terms in the equations involving a combination of oil variables only. With the knowledge of values for the vinegar variables, one can see that such a system will reduce to a set of linear equations in oil variables that can be easily solved. The signing protocol of Rainbow uses multiple such oil-vinegar constructions where each system uses the result of the previous one for its vinegar variable values. The private key essentially contains the coefficients for the equations in these constructions. We can visualize each construction as a *layer* and hence Rainbow is formed by multiple layers. The verification algorithm is a simple substitution of the signature for the variable values in the system of quadratic equations (whose coefficients are contained in the public key). If we obtain the message as the right-hand side, then the signature is considered valid.

Details on Signature Generation: For parameters n and v_1, let GF$(q)^{n-v_1}$ be the message space and GF$(q)^n$ the signature space of Rainbow. Let L_1 : GF$(q)^{n-v_1} \rightarrow$ GF$(q)^{n-v_1}$ and L_2 : GF$(q)^n \rightarrow$ GF$(q)^n$ be two invertible affine linear maps. Furthermore, let F : GF$(q)^n \rightarrow$ GF$(q)^{n-v_1}$ be a function given by a system of $n - v_1$ oil-vinegar polynomials (details are provided later on). The triple (L_1, L_2, F) constitutes the private key of the signature scheme which is kept secret and used to sign a message. The composition $\bar{F} := L_1 \circ F \circ L_2$ is the public key which is used to verify a signature.

In order to generate the signature $Z = (z_1, \ldots, z_n) \in$ GF$(q)^n$ of a given message $Y = (y_{v_1+1}, \ldots, y_n) \in$ GF$(q)^{n-v_1}$ we need to solve the equation

$$L_1 \circ F \circ L_2(Z) = Y \tag{1}$$

for Z. To this end, we first apply L_1^{-1} on Y resulting in $Y' = L_1^{-1}(Y)$ (i.e., we perform a vector addition followed by matrix-vector multiplication). The next step is to solve the *central map equation*

$$F(X) = Y', \tag{2}$$

for $X = (x_1, \ldots, x_n) \in$ GF$(q)^n$. This step is described below in more detail. To obtain the final signature Z we simply apply L_2^{-1} on the solution X.

The central map F of the Rainbow signature scheme consists of $n - v_1$ quadratic oil-vinegar polynomials over GF(q) split into $u - 1$ layers as follows: Let parameters $v_1, \ldots, v_{u+1}$ satisfying $0 < v_1 < v_2 < \ldots < v_{u+1} = n$ be given (*vinegar splitting*). Then Layer $l \in \{1, \ldots, u-1\}$ is defined by the polynomials $p'_{v_l+1}, \ldots, p'_{v_{l+1}}$, where each such polynomial p'_k $(v_l + 1 \leq k \leq v_{l+1})$ is of the form

$$p'_k = \sum_{i<j\leq v_l} \alpha_{ij}^{(k)} x_i x_j + \sum_{i\leq v_l<j<v_{l+1}} \alpha_{ij}^{(k)} x_i x_j + \sum_{i<v_{l+1}} \beta_i^{(k)} x_i \, . \tag{3}$$

The vinegar variables of the polynomials p'_k at layer l are the variables $x_1, \ldots, x_{v_l}$ and the oil variables are $x_{v_l+1}, \ldots, x_{v_{l+1}}$. Note that by evaluating all vinegar variables in p'_k with elements from GF(q), we obtain a linear polynomial. To solve Equation (2), i.e., solving the system of polynomial equations

$$p'_k(x_1, \ldots, x_n) = y'_k, \text{ for } v_1 + 1 \leq k \leq n$$

given $Y' = (y_{v_1+1}, \ldots, y_n)$ we proceed from layer to layer as follows: We start at Layer 1 by assigning random values to the corresponding vinegar variables, i.e., $x_1, \ldots, x_{v_1}$. The first $v_2 - v_1$ polynomial equations reduce to linear equations in the oil variables $x_{v_1+1}, \ldots, x_{v_2}$, cf. (3). By solving this linear sub-system of dimension $v_2 - v_1 \times v_2 - v_1$ using Gaussian elimination, we obtain values for $x_{v_1+1}, \ldots, x_{v_2}$. Thus, we are able to solve the second system of $v_3 - v_2$ linear equations. Continuing this process, we try to iteratively solve the above system of polynomial equations. If the corresponding LSE is not (uniquely) solvable at a particular layer (this has a very low probability for our parameter choices), the process is restarted (new random values for the vinegar variables at the first layer are chosen).

Table 1. Rainbow Parameter Values

Parameter	Rainbow1	Rainbow2
Ground Field	GF(2^8)	GF(2^8)
Message size	24 bytes	28 bytes
Signature size	42 bytes	48 bytes
Number of layers	2	2
Vinegar Splitting	18, 30, 42	20, 34, 48

Design Parameters: A judicious choice of parameter values need to be made for MQ-based constructions to ensure a desired level of security [1], [5], [6], [7]. Two versions of Rainbow are proposed in Table 1, namely Rainbow1 and Rainbow2. Rainbow1 is claimed to provide a security level of 2^{82}. Here F consists of $n - v_1 = 24$ polynomials in $n = 42$ variables split into $u - 1 = 2$ layers using the vinegar splitting parameters $v_1 = 18, v_2 = 30$ and $v_3 = 42$. Rainbow2 is claimed to provide a security level of 2^{90}.

3 Architecture Overview

The overall Rainbow system can be split into three parts. The input/key generation circuitry part will take care of getting the message and generating the needed random numbers for the private key. After this step is complete, the input circuitry shall communicate to the signature generation core part and the signing process will begin. The signature generation core shall also update the input circuitry with its current state, so that it can determine the number of key bytes to be sent to the core. In a particular state of the signature generation step, if certain key ports are unused, the input circuitry shall pad zeroes appropriately for these ports. The output circuitry part shall handle the task of sending the generated signature to the outer world. The design of the input and output blocks is dependent of the bandwidth and the pin constraints of the chip. Since we believe that input-output circuitry can be efficiently implemented, we shall henceforth be concerned only about the design of the signature generation core. The basic operations of signature generation are setting up the linear system of equations and solving them using Gaussian elimination. Setting up the LSE involves matrix-vector multiplication, vector addition and computation of the bilinear form. Among all these aforementioned operations, the most computationally intensive one is Gaussian elimination over $\mathrm{GF}(2^l)$. In the next section, we present G-SMITH, a parallel hardware architecture for fast Gaussian elimination over $\mathrm{GF}(2^l)$. We adapted G-SMITH for the case of Rainbow and designed the datapath architecture. The Rainbow core (shown in Figure 1) is sub-divided into two logical parts: the datapath and the control logic. The control logic sequences the operations and the datapath executes them. The datapath can be further split into two parts, namely the LSE setup and the LSE solving stages. The G-SMITH hardware is used for solving LSEs. It is reused to perform matrix-vector multiplication, vector-addition and the computation of the bilinear form during the time when it does not perform LSE solving. These operations are needed for computing the L_1^{-1} and L_2^{-1} mapping and the setting-up of linear equations.

4 Solving LSEs using G-SMITH

In order to obtain a high-performance signature generation architecture for Rainbow, the development of a fast parallel hardware architecture for the Gaussian elimination step is required. In [3] the authors propose an efficient architecture implementing Gauss-Jordan elimination over GF(2). In this paper we extend the results to $\mathrm{GF}(2^l)$.

Proposed Hardware Algorithm: The main idea for our hardware-based Gauss-Jordan elimination is to perform *normalization* operations (multiplying the inverse of the pivot element with all other elements of the pivot row) and the *elimination* operations (adding multiples of the normalized pivot row to the other rows) all in parallel. Gaussian elimination over $\mathrm{GF}(2^l)$ transforms to Algorithm 1 which

Algorithm 1 Hardware-based Gauss-Jordan over $\mathrm{GF}(2^l)$

```
Require: Regular matrix $A \in \mathrm{GF}(2^l)^{n\times n}$
1: for each column $k = 1 : n$ do
2:     while $a_{11} = 0$ do
3:         $A := shiftup(n-k+1, A)$;
4:     $\vec{a}_1 := normalize(\vec{a}_1)$
5:     $A := eliminate(A)$;
```

works as described in the following.

Let $\vec{a}_i$ denote the i-th row vector of A. In the k-th iteration a pivot element is obtained by shifting[1]

$$shiftup : \{1,\ldots,n\} \times \mathrm{GF}(2^l)^{n\times n} \to \mathrm{GF}(2^l)^{n\times n}$$
$$(i, (\vec{a}_1,\ldots,\vec{a}_n)^T) \mapsto (\vec{a}_2,\ldots,\vec{a}_i,\vec{a}_1,\vec{a}_{i+1},\ldots\vec{a}_n)^T$$

[1]In the actual hardware implementation we keep track of the used rows by means of a $used$-flag instead of using a counter.

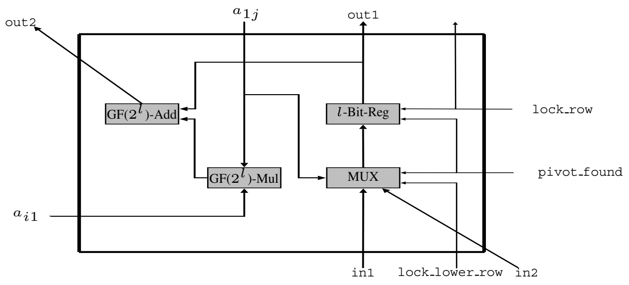

Figure 2. A basic mesh cell storing a_{ij}.

until a_{11} is a non-zero element. After a pivot element has been found in the k-th iteration, we normalize the first row:

$$normalize : \mathrm{GF}(2^l)^n \to \mathrm{GF}(2^l)^n$$
$$(a_{i1}, a_{i2}, \ldots, a_{in}) \mapsto (1, a_{i2}\cdot a_{11}^{-1}, \ldots, a_{in}\cdot a_{11}^{-1})$$

Then we add the first row $\vec{a}_1$ multiplied by a_{i1} to all other rows $\vec{a}_i$ where $i \neq 1$ and $a_{i1} \neq 0$ to eliminate these elements. In addition, we do a cyclic shift-up of all rows and a cyclic shift-left of all columns. By doing the cyclic shift-up operations after an elimination, rows already used for elimination are "collected" at the bottom of the matrix, which ensures that these rows are not involved in the pivoting step anymore (elimination, cyclic shift-up and cyclic shift-left):

$$eliminate : \mathrm{GF}(2^l)^{n\times n} \to \mathrm{GF}(2^l)^{n\times n}$$
$$\begin{pmatrix} 1 & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix} \mapsto \begin{pmatrix} a_{22}\oplus(a_{12}a_{21}) & \ldots & a_{2n}\oplus(a_{1n}a_{21}) & 0 \\ \vdots & & \vdots & \vdots \\ a_{n2}\oplus(a_{12}a_{n1}) & \ldots & a_{nn}\oplus(a_{1n}a_{n1}) & 0 \\ a_{12} & \ldots & a_{1n} & 1 \end{pmatrix}$$

$shiftup$, $normalize$, and $eliminate$ can be computed within a single clock cycle in hardware. We always have n applications of $normalize$/$eliminate$. Additionally, an expected number of $n2^{-l}$ applications of $shiftup$ need to be executed if the matrix elements are uniformly distributed over $\mathrm{GF}(2^l)$, as in our case. Thus, for $n = 12$ and $l = 8$ executing the algorithm requires about 12.05 clock cycles.

Functional Description: In order to implement Algorithm 1 in hardware, we use a mesh structure (rectangular cell array): the whole device consists of four different types of memory cells, which realize the three basic operations of the algorithm. The design at hand comprises a parallel implementation of the operations where the output of a zero-tester applied to the pivot element is used as multiplexing signal for the appropriate result.

Mesh Cells and Their Interconnections: Our mesh architecture consists of four types of cells: the cell in the upper left corner of the mesh structure is called the pivot cell, the cells in the first row resp. first column unlike the pivot cell are called the pivot row resp. pivot column cells, all other cells are called basic cells. Here we describe the more complex basic cells only.

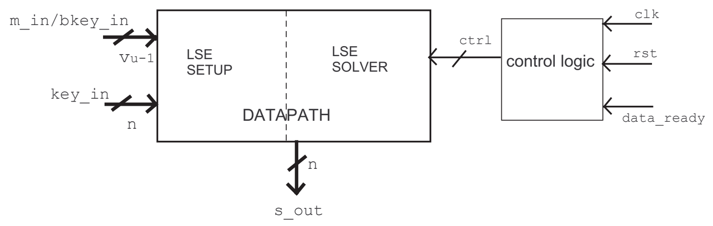

Figure 1. Rainbow Signature Generation Core

The design and interconnections of a *basic cell* are depicted in Figure 2. Each cell stores one element $a_{ij} \in \mathrm{GF}(2^l)$ of the matrix in its l-bit register and contains one adder and one multiplier to implement the *eliminate*-operation. It has *local* connections to its 4 direct neighbors and some *global* connections. Considering Figure 2 we distinguish between two types of connections, namely data connections (thick lines) which are l-bit busses for exchanging elements from $\mathrm{GF}(2^l)$ between the cells and 1-bit control signal connections (thin lines). As to the local connections, a cell has a data and a control connection (`out1` and `lock_row`) to its upper neighbor, a data and a control connection (`in1` and `lock_lower_row`) to its lower neighbor, and only data connections (`out2` resp. `in2`) to its upper left and lower right neighbor in the mesh. Note that a cell receives inputs only from its lower direct neighbors and sends output only to its upper direct neighbors. The mesh is wrapped around in the sense that the lower neighbours of cells in the last row are the respective cells of the first row. Furthermore, a cell is connected to a global network: the global network comprises a control signal (`pivot_found`) from the pivot cell, a data signal (a_{i1}) from the first cell of the respective row (a pivot column cell) and a data signal (a_{1j}) from the first cell of the respective column (a pivot row cell) the considered cell belongs to. Moreover, the *used*-flag for the actual row is provided by the global control signal `lock_row`. Besides the adder and the multiplier that are used to implement the *eliminate*-operation, a basic cell also contains a multiplexer (MUX). By means of this MUX which receives the signals `pivot_found` and `lock_lower_row` as input we control which data is input to the register depending on the operation that is currently executed. Whether this data is actually stored in the register is controlled by the signals `pivot_found` and `lock_row`.

In the *pivot cell* we do not need any multiplier or adder but an inverter and a zero tester. The *pivot row cells* are equipped with a multiplier but no adder. They receive the output of the inverter of the pivot cell. The output of the multiplier of such a cell is connected to all basic cells of the respective column j and provides the global data signal a_{1j}. A *pivot column cell* contains neither an adder nor a multiplier. Its register is connected to all basic cells of the respective row i and provides the global data signal a_{i1}.

Hardware Complexity of G-SMITH over GF(2^l): The G-SMITH architecture for solving systems of linear equations over $\mathrm{GF}(2^l)$ is basically comprised of the following elements: $n(n+1)$ l-bit registers, one $\mathrm{GF}(2^l)$-inverter for computing a_{11}^{-1}, n $\mathrm{GF}(2^l)$-multipliers for normalizing the first row, and $n(n-1)$ $\mathrm{GF}(2^l)$-multipliers and adders for the elimination step.

Adaptions for Multivariate Signing: The G-SMITH architecture is used for solving the linear system of equations. The maximum value taken by the difference between the vinegar variables in any two subsequent layers determines the size of G-SMITH. Some modifications are required in order to enable the reuse of G-SMITH during operations other than Gaussian elimination. A `Gsel` signal was added to all cells, in order to choose between Gaussian elimination and other operations. Most of the adders and multipliers are reused for vector addition, matrix-vector multiplication and computation of bilinear form. It should be noted that despite the area overhead due to the addition of multiplexers, the reuse of G-SMITH cells helps to perform the LSE generation process in parallel, thereby drastically reducing the computation time.

5 Rainbow Datapath

The detailed datapath of Rainbow is depicted in Figure 3. The thick lines are data lines. The control signals for all the blocks come from the main control logic. The `gsmith_out` bus is de-multiplexed to `gsmith_out1` and `gsmith_out2` buses, using a control signal `k0` (depending on the layer). This is not shown in the figure. The cells in G-SMITH that have a left slash are re-used by the Vector Multiply Add unit and the ones with the right slash are reused by the Bilinear Form unit.

Datapath Hardware Cost: The area required by the overall architecture is measured in terms of *gate equivalents* (GE). The strategy for estimating the total hardware cost is as follows: we have re-used the multipliers, adders and registers from G-SMITH for other parts of the signature generation process. As a result, we have added several 8-bit 2:1 muxes to enable re-use. First, we estimated

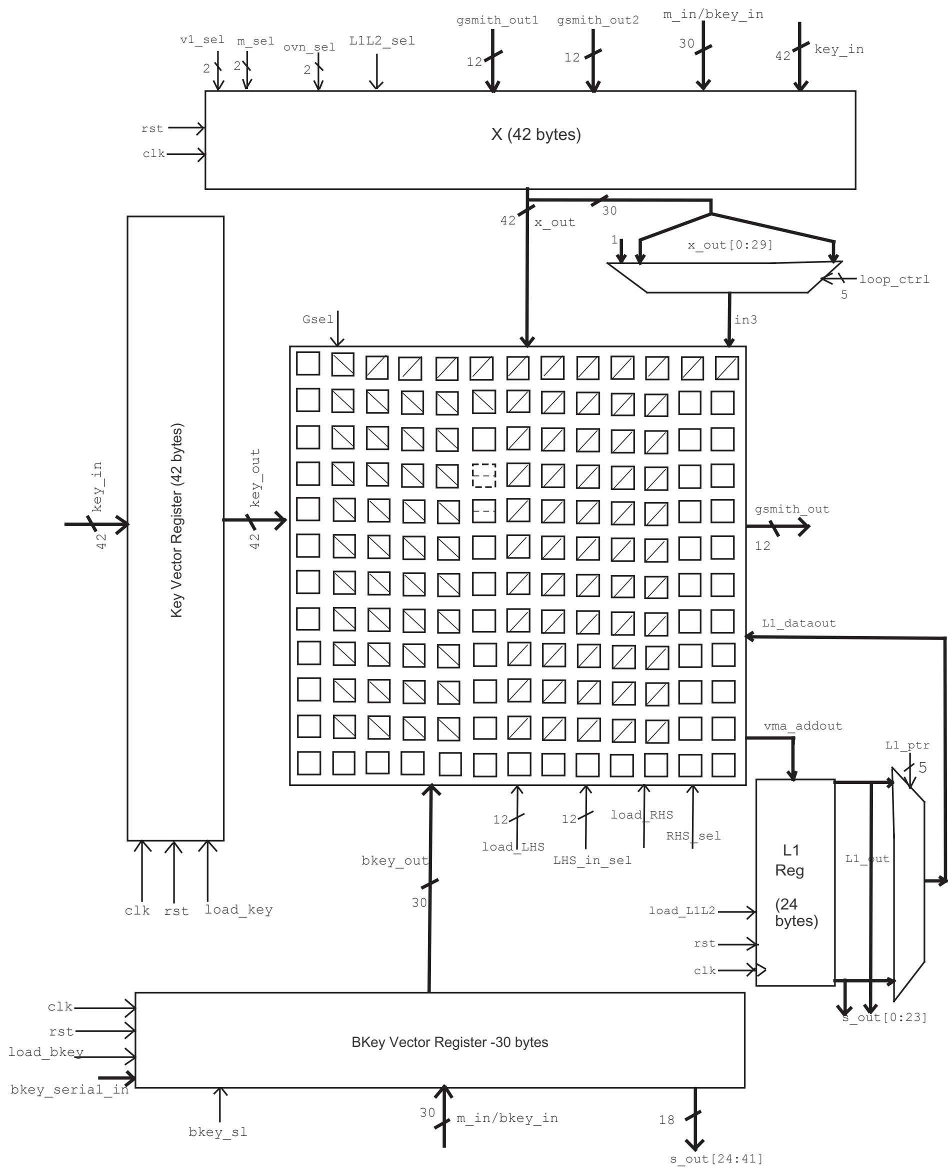

Figure 3. Rainbow detailed datapath

the number of such muxes added to the design. This count was calculated to be around 821 and 968 for Rainbow1 and Rainbow2 respectively. Then, we estimated the additional hardware, apart from G-SMITH in the datapath. Finally, we summed up all these to get the overall GE. The total GE for the Rainbow1 and Rainbow2 were estimated to be 66874.25 and 85272.45 respectively.

Running Time: Since one element of the resultant vector is calculated in a single cycle from the matrix-vector multiplier unit, the total number of cycles required for the linear transformations L_1^{-1} and L_2^{-1} is $2n - v_1 + 4$. The number of cycles required to setup the linear equations over $u - 1$ layers can be estimated using the formula $\sum_{i=1}^{u-1}(v_{i+1} - v_i) \cdot (v_i + 6)$. So, the total number of cycles for Rainbow1 is estimated to be ≈ 814 cycles.

6 Implementation

This section explains the methodology used to implement the Rainbow datapath and the control logic finite state machine. We also provide the ASIC implementation details and results for our design[2].

[2]For the purpose of functional simulation and verification, we used the *Synopsys Scirocco Virtual Simulator environment* version *X-2005.06*. The

Atomic blocks: The basic building blocks for the architecture are the GF(2^8) adder, multiplier and inverter. There are three popular bases of representation of GF(2^8) data from hardware point of view namely, standard base (SB), normal base (NB) and dual base (DB). We have chosen the standard base representation, since it does not involve any overhead in conversion. Area-efficient arithmetic architectures for the standard base multiplier and inverter were proposed in [8, 9]. We have used these blocks for our atomic operations in the algorithm.

G-SMITH: We implemented the G-SMITH datapath and controller which was presented in Section 4. The architecture was programmed in VHDL and is kept generic for easily changing the matrix dimensions. The area for the cryptographically interesting matrix size ($n = 12$) is 42366.33 GE. The initial timing reports indicate that the system can be clocked upto a maximum frequency of 93 MHz.

Rainbow: The total number of clock cycles required for signature generation with Rainbow (including L_1^{-1}, F^{-1} and L_2^{-1}) is 804. The core was synthesized and the area was measured to be 63592.88 GE. Timing reports show that the core can be clocked upto a maximum frequency of 67 MHz. This would translate to a signature generation time of 0.012 ms.

7 Comparison

For the benchmarks the well-established RSA (with a 1024-bit modulus) and elliptic curves (binary fields of about 160 bits[3]) signature algorithms are taken as well as the en-TTS scheme. The performance of these implementations is very close to the best known ones. The comparison results are given in Table 2. Although different process technologies and tools were used to implement these schemes, our comparision is fair since it is done at the architectural level, where area is measured in gate equivalents and time is calculated using the reported frequency. The performance metric of our Rainbow implementation is significantly better than that of the considered RSA, ECDSA and en-TTS implementations.

References

[1] J. Ding and D. Schmidt, "Rainbow, a new Multivariable Polynomial Signature Scheme," in *Proc. of ACNS'05*, 2005.

architecture was synthesized for the Illinois Institute of Technology (IIT) standard cell library based on the AMI 0.35μ technology. *Synopsys Design Compiler* version *X-2005.09* was used for synthesis.

[3]No implementations for special elliptic curves over special finite fields have been treated, which can potentially somewhat reduce the time-area product.

Table 2. Comparison of Rainbow1 with other signature schemes

Scheme	Area(GE)	Freq.(MHz)	Time(ms)	Time$\times$area*
RSA [10]	250000	200	1.74	570.02
ECDSA [11]	117500	510	0.19	29.25
ECDSA [12]	191000	20	4.4	1100.74
en-TTS [4]	21000	67$^{\#}$	0.9	24.76
Rainbow1	63593	67	0.012	1.00

* The time-area products are normalized to Rainbow.
$^{\#}$ The ASIC design for en-TTS [13] is targeted at low-cost RFID applications and the maximal frequency of only 100 kHz is specified for it. However, our initial analysis of the hardware architecture indicates that it can be clocked at much higher frequencies. For comparison purposes we assume that the clock frequency for the design is 67 MHz.

[2] A. Kipnis, J. Patarin, and L. Goubin, "Unbalanced Oil and Vinegar signature schemes," in *Proc. of EUROCRYPT'99*, 1999.

[3] A. Bodganov, M. Mertens, C. Paar, J. Pelzl, and A. Rupp, "A Parallel Hardware Architecture for fast Gaussian Elimination over GF(2)." in *Proc. of FCCM'06*, 2006.

[4] B.-Y. Yang, C.-M. Cheng, B.-R. Chen, and J.-M. Chen, "Implementing Minimized Multivariate PKC on Low-Resource Embedded Systems," in *SPC'06*, 2006.

[5] O. Billet and H. Gilbert, "Cryptanalysis of Rainbow," in *Proc. of SCN'06*, 2006.

[6] L. H. J. Ding, B. Yang and J. Chen, "Notes on Design Criteria for Rainbow-Type Multivariates," IACR eprint report 2006/307, 2006.

[7] B.-Y. Yang and J. Ding, "New Differential Attacks on Rainbow-like Multivariate Signature Schemes and Improved Designs," Preprint, University of Cincinnati, 2007.

[8] C. Paar, "Efficient VLSI architectures for bit-parallel computation in Galois Fields," Ph.D. dissertation, Institute of Expermiental Mathematics, University of Essen, Germany, 1994.

[9] C. Paar and M. Rosner, "Comparison of Arithmetic Architectures for Reed-Solomon decoders in reconfigurable hardware," in *Proc. of FCCM '97*, 1997.

[10] J. Großschädl, "High-Speed RSA Hardware Based on Barret's Modular Reduction Method," in *Proc. of CHES'00*, 2000.

[11] A. Satoh and K. Takano, "A scalable dual-field elliptic curve cryptographic processor," *IEEE TC*, 2003.

[12] R. Schroeppel, C. Beaver, R. Gonzales, R. Miller, and T. Draelos, "A Low-Power Design of an Elliptic Curve Digital Signature Chip," in *Proc. of CHES'02*, 2002.

[13] B.-Y. Yang and J.-M. Chen, "Building secure tame-like multivariate public-key cryptosystems: The new TTS," in *Proc. of ACISP'05*, 2005.