Fruit: Ultra-Lightweight Stream Cipher with Shorter Internal State

Vahid Amin Ghafari, Honggang Hu
Key Laboratory of Electromagnetic Space Information, Chinese Academy of Sciences,
University of Science and Technology of China, Hefei, China, 230027
vahidaming@mail.ustc.edu.cn, hghu2005@ustc.edu.cn

Abstract

In eSTREAM project, a few lightweight stream cipher for hardware was introduced (2008) and then in FSE 2015 Sprout was proposed. Sprout introduced a new idea, design of stream cipher with shorter internal state by using key not only in initialization but also in keystream generation, but it was insecure. Grain-v1 is almost the lightest secure cipher in world. Fruit stream cipher is successor of Grain and Sprout stream ciphers that we show is secure and ultra-lightweight cipher. The size of LFSR and NFSR in Fruit is only 80 bits (for 80-bit security), while for resistance stream cipher against Time-Memory-Data trade-off attack, internal state should be at least twice of security level. For compensate of this we use some new ideas in design.

Keywords: Stream Cipher, Ultra-lightweight, Lightweight, Grain, Sprout, Cryptographic Primitive, NFSR, LFSR, Hardware Implementation

Introduction

Nowadays the need to secure lightweight symmetric cipher is obviously more than eSTREAM project time (this is provable by a lot of paper in design and cryptanalysis of light cipher). WSN and RFID and mobile phone are instances lead us to accept important of design new and secure lightweight ciphers.

Three Stream ciphers have been introduced in the hardware profile of portfolio of eSTREAM project. They are Trivium and MICKEY 2.0 and Grain-v1 [7, 8, 9]. Grain-v1 use NFSR and LFSR together, the linearity section would guarantee good statistical properties and guarantee about period, while the nonlinearity part would protect against attacks that can be mounted against a linear cryptosystem. Grain-128 was introduced in

2006 [10] and some attacks proposed to it [11, 12, 13, 14, 15, 16, 17]. Indeed Grain-128 was not secure as expected (such as Grain-v1). Grain-128a [18] was proposed in 2011. Although some attacks have been applied to Grain-128a [19, 20, 21], due to practicality point of view, still it is stand-up.

Sprout is stream cipher with shorter internal state that was introduced in FSE 2015 [1]. A short while after Sprout was introduced many attacks was published against it [3, 6, 4, 23, 2, 5]. Although it has become clear that Sprout is insecure, but it had a great new idea that help to design stream cipher with smaller area size. The new idea is to effect secret key not only in initialization but also during key generation. Actually this idea help to extend internal state to secret key. Due to in the most of application we should save key for reuse by different IV, the idea help to us have bigger internal sate (therefor we can design stronger ciphers). On the other hand, we need to save key in a fix memory in some applications (in this case one fix key is sufficient for ever) and it is known that save fix bits need less area size in compare to save bits in temporary memory (e.g. bourn fix key in fuse). Thus we can design stream ciphers with shorter internal state [1]. Fix key was not used in a suitable way in the design of Sprout and in the papers about cryptanalysis of Sprout mentioned that design of stream cipher (by the new idea) is too hard [3, 5] and in other paper, authors told it is fascinate [23] and other authors predict very soon secure cipher will be proposed by this new idea [2].

Necessary condition for stream ciphers to be resistance against time memory data trade-off attack is internal state size that should be at least twice of his security (while secret key only use in initialization round). For example in Trivium and MICKEY 2.0 and Grain-v1, portfolio of eSTREAM project, we can see this fact. But we know that in some application (such as RFID, WSN) there are less resource and we need new stream cipher with minimal internal sate. We think that this is new generation in design of stream cipher and we define ciphers with less than 950 GE (gate equivalents) ultra-lightweight stream ciphers (Sprout is ultra-lightweight stream cipher).Here we propose another reduced internal state stream cipher, Fruit, that is successor of Grain and Sprout that we will show is secure and ultra-lightweight cipher. Really in nature after a while grain becomes sprout and in finally we have fruit. We think that Grain and Sprout were new generation in design of stream ciphers and now Fruit is mature in stream ciphers.

We summarize our new idea in design of Fruit as follow:

1- New round key function (most weaknesses of sprout is related to round key function)
2- New nonlinear feedback for NFSR and feedback for LFSR and output function (in Sprout they are similar to Grain-128a, they are different only in positions)
3- Increase the size of LFSR to achieve longer output keystream in each loading (in compare to Sprout)

4- New way to load IV in initialization (in Grain family and Sprout IV load in LFSR and NFSR)

In continue of the paper, we explain about design of Fruit and next design criteria. Then we show Fruit is resistance to known attacks. Finally we tell about hardware implementation of Fruit.

Design Details of Fruit

Internal state consists of 43-bit LFSR ($l_t, \dots, l_{t+42}$), 37-bit NFSR ($n_t, \dots, n_{t+36}$), 15-bit counter, C ($c_t^0, \dots, c_t^{14}$). Inputs for cipher are 80-bit secret key, K (k_i ,$0 \leq i \leq 79$) and 80 bits public Initial Value, IV (v_i ,$0 \leq i \leq 69$). Note that maximum number of stream bits that can be produced from one key and IV is 2^43 bits and every key should use less than 2^15 times with different IVs. It is not acceptable to reuse IV, i.e. use identical IV with different keys and IV should produce in random way.

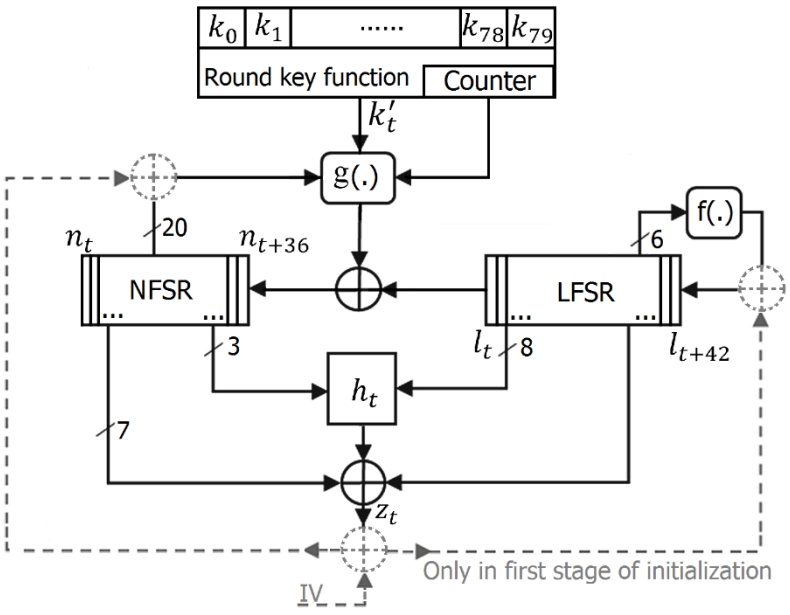

Fig. 1: Block Diagram of Fruit

Now we explain every part of cipher in detail:

-Counter: first 7 bits of counter are allocated to round key function and last 8 bits are allocated to initialization and keystream production. These two sections work (count) independently, i.e. first part is increased one by one in every clock (i.e. the value of $c_t^0, \dots, c_t^6$ increases in each clock one unit) and second part ($c_t^7, \dots, c_t^{14}$) count from zero to 255 independently. Actually we have two counters and we have two increases in both of

them at each clock. We need our counter work continually, i.e. after first and second part become all ones, again count from zeroes to all ones. Note that c_t^6 and c_t^{14} are LSB of two counters, i.e. before first clock our counter is (000000000000000) and then after first clock is (000000100000001).

-**Round key function**: we define $s = (c_t^0 c_t^1 c_t^2 c_t^3 c_t^4 c_t^5)$, $y = (c_t^3 c_t^4 c_t^5)$, $z = (c_t^4 c_t^5 c_t^6)$, $p = (c_t^0 c_t^1 c_t^2 c_t^3 c_t^4)$, $q = (c_t^1 c_t^2 c_t^3 c_t^4 c_t^5)$, and $r = (c_t^3 c_t^4 c_t^5 c_t^6)$. We combine 6 bits of key to obtain bits of round key as fallow in each clock:

$$k_t' = k_s . k_{y+64} \oplus k_{z+72} . k_p \oplus k_{q+32} \oplus k_{r+64}$$

-g function: we use one bit from LFSR and 1 bit of the counter and k_t' and 20 bits of NFSR as a variables of g function for clocking of NFSR.

$$n_{t+37} = k_t' \oplus l_t \oplus c_t^{10} \oplus n_t \oplus n_{t+10} \oplus n_{t+20} \oplus n_{t+12} . n_{t+3} \oplus n_{t+14} . n_{t+25} \oplus n_{t+8} . n_{t+18} \oplus n_{t+5} . n_{t+23} . n_{t+31} \oplus n_{t+28} . n_{t+30} . n_{t+32} . n_{t+34}$$

-f function: feedback function in LFSR is primitive. Thus maximum string will be produced by following function.

$$l_{t+43} = l_t \oplus l_{t+8} \oplus l_{t+18} \oplus l_{t+23} \oplus l_{t+28} \oplus l_{t+37}$$

-h function: this function produce pre-output stream from LFSR and NFSR sates and key.

$$h_t = n_{t+1} . l_{t+15} \oplus l_{t+1} . l_{t+22} \oplus n_{t+35} . l_{t+27} \oplus n_{t+33} . l_{t+11} \oplus l_{t+6} . l_{t+33} . l_{t+42}$$

-running key: output will be produced by 7 bits from NFSR and 1 bit from LFSR and output of h function.

$$z_t = h_t \oplus n_t \oplus n_{t+7} \oplus n_{t+13} \oplus n_{t+19} \oplus n_{t+24} \oplus n_{t+29} \oplus n_{t+36} \oplus l_{t+38}$$

-Initialization of cipher: we extend IV to 80 bits by concatenating 9 bits zero and 1 bits one to the first of IV, as follow.

$$IV' = 1000000000 v_0 v_1 v_2 \dots v_{67} v_{68} v_{69}$$

In initialization round, key bits is loaded to NFSR and LFSR respectively from LSB to MSB (k_0 to n_0 , k_1 to n_1 …and k_{37} to l_0 , k_{38} to l_{41} …). We put $c_0^0 = c_0^1 = \cdots = c_0^{13} = c_0^{14} = 0$ and then, we clock 80 times the cipher and before each clock, XOR output with IV' bits and also feedback output stream to NFSR and LFSR,

i.e. $z_i \oplus v'_i$, $0 \leq i \leq 79$ (as show in fig. 1). Now, we put all bits of first section of counter equal to LSB of NFSR except last bit of the first section of counter that is equal to LSB of LFSR ($c_{80}^0 = n_{80}, c_{80}^1 = n_{81}, \dots, c_{80}^4 = n_{84}, c_{80}^5 = n_{85}, c_{80}^6 = l_{80}$) and also we put $l_{80} = 1$ for preventing all zeroes in LFSR.
Then cipher should clock 80 times without feedback output in LFSR and NFSR (i.e. during last 80 clocks we disconnect feedback of z_t to LFSR and NFSR). Thus, the cipher doesn't produce any output stream in the 160 initial clocks, i.e. we discard z_0 to z_{159}. Now cipher is ready to produce first bit of keystream, i.e. z_{160}.

Design criteria

-Limitation for producing output stream: due to length of LFSR we can produce 2^43 bits in maximum in every initialization (period of NFSR is at least 2^43 bits). We think that 1 terabyte is almost sufficient for all application because our cipher is special for hardware application (e.g. WSN and RFID).

-Round key function: we produce 2^7 different key from original key. Attacker can (with guessing internal states and known output keystream) obtain some bits of k'_t, but due to the unknown counter (unknown index of key in round key function), it is not to easy solve equations system. We define round key function so that involves bits of key in output. Round key function of Sprout cipher was designed so that in some clocks none of key bits involves in the output or in g function.

-g function: The function that produces n_{t+37} is in 20 variables of NFSR due to light implementation in hardware in compare to Grain-v1 and Sprout. If we suppose $k'_t \oplus c_t^{10} \oplus l_t = 0$, the nonlinearity of g function will be $2^3 \times 3760$ and resiliency 2. Variables for high degree term is chosen from n_t with $t > 27$ that cause the degree of variables very soon hit maximum possible degree in NFSR.

-f function: feedback polynomial is primitive, so the period of produced string by LFSR with non-zero initial is maximum. Due to after disconnecting feedback of output bit to LFSR, we put l_0=0, we are sure that period of LFSR and NFSR is at least 2^43 bits. Some attacks was proposed to Grain and Sprout from this weakness [22, 23].

-Output function: The nonlinearity of h function is 976. We add 8 linear terms in order to increase nonlinearity to 249856 and also to make function with 7 resiliency. The best linear approximation of output function has 8 terms with $2^{-5.415}$ bias.

Note that we use n_{t+36} and $n_t$in output function for preventing produce keystream in next clock and previous clock with unknown k'_t.

Resistance against known attacks

Security level of fruit is 80 bits, thus here we discuss about the feasibility of applying some main attacks to Fruit.

-Linear Approximation Attack: this attack was applied to Grain-v0 [24]. In [24] discussed that if NFSR and output function choose with high nonlinearity and suitable resiliency, it will be strength to Linear Approximations attack. We choose NFSR and output function with high nonlinearity and good resiliency and also in Fruit a nonlinear function of key is involved on NFSR. If attacker obtain linear approximation of output and also linear approximation of g function and then eliminate NFSR bits between two relations, he can obtain follow relation with $2^{-43.86}$ bias.

$$z_t \oplus z_{t+10} \oplus z_{t+20} \oplus z_{t+80} = l_t \oplus k'_t \oplus c_t^{10} \oplus l_{t+7} \oplus k'_{t+7} \oplus c_{t+7}^{10} \oplus l_{t+13} \oplus k'_{t+13} \oplus c_{t+13}^{10} \oplus l_{t+19} \oplus k'_{t+19} \oplus c_{t+19}^{10} \oplus l_{t+24} \oplus k'_{t+24} \oplus c_{t+24}^{10} \oplus l_{t+29} \oplus k'_{t+29} \oplus c_{t+29}^{10} \oplus l_{t+36} \oplus k'_{t+36} \oplus c_{t+36}^{10} \oplus l_{t+38}$$

Now, if attacker suppose that round key bits and counter bits are equal to zero, and if he try to obtain a relation only based on output bits (by using feedback polynomial of LFSR), the bias of relation is $2^{-301.02}$. This bias is too small and therefor our cipher is strength to this attack.

-Guess and determine attack: due to shorter internal sate in Sprout and Fruit, this attack is very important to us (Sprout was weak against this attack [23]). If attacker guess all bits of internal sate in the Sprout, he can clock 2 times forward and one time backward (with unknown key) and in every clock he can decrease the wrong candidates of internal sate. In the next clocks attacker obtain one bit of key or decrease the wrong candidates of internal sate. We strengthen round key function and use n_{t+36} and n_t in output function for preventing produce keystream in next clock and previous clock with unknown key. If attacker can obtain

some bits of k'_t, due to unknown number of first part of counter (unknown index of key in round key function) it is too hard which attacker can solve equations and obtain key.
If attacker guess all bits of counter and LFSR and NFSR, i.e. 87 bits, he can obtain round key function bits. In this situation due to every bit of round key function is dependent on some bits of key, it is not possible that attacker can identify wrong candidates of internal states before 80 clocks (except in the first clock he can identify half of wrong candidates). The computational complexity of this attack (at least) is 80.2^{86} that is more than complexity of exhaustive attack, i.e. 2^{80}. Thus, Fruit is resistant against this attack.

-Time-Memory-Data Trade-off Attack: in literature is famous that if the size of internal state is not at least twice of security level, cipher is weak to this attack. In Fruit we use fix key as an internal state and also some bits of counter in round key function and also we independently use k'_t in g function to preventing key bypass in g function (such as attack to Sprout [2]), therefor there is no problem from this view.

-Related-key Attack: There is weakness in initialization process of Grain-128a [21] and Sprout [3]. Designers of Sprout ruled out related key attack. They believe that due to key is fix in ultra-lightweight ciphers, this attack is not workable on Sprout [1]. Nevertheless we propose new scheme in initialization round to strength against this attack. We do not load IV bits directly in internal sate and so do not combine IV and key bits straightforward together. Also we use asymmetric padding with IV, thus we are sure there is no weakness to this attack.

-Cube attack: due to the suitable clock number, it is too hard to find any low degree multiplicative expression (of some bits of IV) base on key in Boolean function of output. Therefor our design is resistant against to all type of Cube attack.

-Algebraic attack: this attack have not been applied to Grain family but combination of this attack was applied to Sprout [4]. Short internal state (or we can tell weak round key function) in Sprout has been caused this weakness. We strengthen round key function and involve bits of key independently in g function, so we think Fruit is secure against this attack. Due to fast growth of degree in the internal state of Fruit, it is not possible for attacker to apply pure algebraic attack. But here we show that a combine of guess and determine attack and algebraic attack is not applicable on Fruit.

If attacker guess bits of NFSR and bits of counter and bits of round key function, then he can obtain two equations in every clock (one from output and one from round key function). These equations are degree 2 and it is not easy to salve, but we suppose that attacker can salve equations of output and obtain in every clock 1 bits of LFSR. In this scenario attacker should guess 40 bits of round key function. Totally attacker should guess $37 + 7 + 43$ that is more than computational complexity of exhaustive attack.

Hardware implementation cost

Design of lightweight cipher is very important in industries while we need to light ciphers in many filed such as communication and WSN and RFID and etc. due to stream ciphers are lighter than block ciphers and very lighter than public ciphers, it is obvious that design of ultra-lightweight stream cipher is very important. Our goal was design strength cipher with less than 950 GE. We use gate count for fair compare of hardware implementation cost between Grain-v1 and Sprout and Fruit proposed in Grain family and Trivium cipher [7, 10, 18]. Due to hardware implementation cost is dependent on many factors such as hardware used in implementation and techniques of implementation and etc., and also due to it is not possible in clear way to show provable results of implementation, we compare gate count of ciphers according to Table 1 [7, 10, 18]. Note that this compare is not consist of all parts such as multiplexers needed in order to switch between key/IV loading, initialization, but it consist of main parts of every ciphers. We don't dedicate any gate to key bits, because in every cipher should save key for reuse with different IVs.

Table 1. The gate count used for different functions

Function	Gate Count
NAND2	1
NAND3	1.5
NAND4	2
XOR2	2.5
Flip flop	8

Table 2. The compare of estimated gate count for implementation of Grain-v1 and Sprout and Fruit

	Grain-v1	Sprout	Fruit
LFSR	8*80	8*40	8*43
NFSR	8*80	8*40	8*37
f function	2.5*5	2.5*5	2.5*5
g function	75	56.5	31.5
Output	53	38	38
Round key function	0	85.5	65.5
etc.	64#	0	64#
Total gate count	1484.5	832.5	851.5

#: Related to counter in initialization round

Conclusion

Fruit stream cipher in compare with Grain-v1 is very lightweight in hardware implementation and due to Grain-v1 is the lightest candidate in eSTREAM project, it is obvious that design of secure stream ciphers such as Fruit is very interesting. We showed that Fruit unlike Sprout cipher is secure with some new ideas. In most application of symmetric cipher secret key (for reuse with other IVs) should save (in a memory) and here we show how we can exploit it in the design. We showed that suitable use of fixed secret key as an internal state cause to save in area size while we can design secure cipher. We hope the design of Fruit (as a first secure ultra-lightweight stream cipher) will be beginning of the design of ultra-lightweight ciphers.

Acknowledgement

This work has been supported by CAS-TWAS President's Fellowship for International PhD program.

References

[1] Armknecht, F., Mikhalev, V.: On lightweight stream ciphers with shorter internal states. Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 451–470. Springer, Heidelberg (2015)
[2] Esgin, M.F., Kara, O.: Practical Cryptanalysis of Full Sprout with TMD Tradeoff Attacks. Selected Areas in Cryptography - SAC 2015. LNCS, vol. 9566 pp 67-85. Springer, Heidelberg (2015)
[3] Hao, Y.: A Related-Key Chosen-IV Distinguishing Attack on Full Sprout Stream Cipher. http://eprint.iacr.org/2015/231.pdf
[4] Maitra, S., Sarkar, S., Baksi, A., Dey, P.: Key Recovery from State Information of Sprout: Application to Cryptanalysis and Fault Attack. http://eprint.iacr.org/2015/236.pdf

[5] Zhang, B. ,Gong, X. : Another Tradeoff Attack on Sprout-Like Stream Ciphers, ASIACRYPT 2015, Part II, LNCS 9453, pp. 561–585, 2015
[6] Lallemand, V., Naya-Plasencia, M.: Cryptanalysis of full sprout. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology – CRYPTO 2015. LNCS, vol. 9215, pp. 663–682. Springer, Heidelberg (2015)
[7] D e Can n i`ere, C., Preneel, B.: Trivium. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 244–266. Springer, Heidelberg (2008)
[8] Babbage, S., Dodd, M. "The Stream Cipher MICKEY 2.0" ECRYPT Stream Cipher Project Report. http://www.ecrypt.eu.org/stream/ p3ciphers/mickey/mickey_p3.
[9] Hell, M., Johansson, T., Meier, W. "Grain—A Stream Cipher for constrained environments" New Stream Cipher Designs, 2008 Lecture Notes in Computer Science, vol. 4986, pp. 179–190, available at http://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain_p3.pdf.
[10] Hell, M., Johansson, T., Maximov, A., Meier, W. "A Stream Cipher Proposal: Grain-128," in International Symposium on Information Theory—ISIT 2006. IEEE, 2006
[11] Küçük, Ö, "Slide resynchronization attack on the initialization of Grain¨1.0," eSTREAM, ECRYPT Stream Cipher Project, Report 2006/044, 2006, http://www.ecrypt.eu.org/stream.
[12] Dinur, I., Shamir, A. "Breaking grain-128 with dynamic cube attacks," FSE 2011, Lyngby, Denmark, 2011, pp. 167–187.
[13] Dinur, I. ,Güneysu, T., Paar, C., Shamir, A., Zimmerman, R. "An experimentally verified attack on full grain-128 using dedicated reconfigurable hardware," Proc. ASIACRYPT 2011, Seoul, South Korea, 2011, pp. 327–343.
[14] J. M. Miodrag, G. Sugata, P. Goutam, and I. Hideki, "Generic cryptographic weakness of K-normal Boolean functions in certain stream ciphers and cryptanalysis of grain-128," Periodica Mathematica Hungarica, vol. 65, no. 2, pp. 205–227, Dec. 2012.
[15] J. P. Aumasson, I. Dinur, L. Henzen, W. Meier, and A. Shamir, "Efficient FPGA implementations of high-dimensional cube testers on the stream cipher grain-128," IACR Cryptology ePrint Archive, pp.218–218, 2009
[16] P. Stankovski, "Greedy distinguishers and nonrandomness detectors," INDOCRYPT 2007, Hyderabad, India, 2010, pp. 210–226.
[17] S. Knellwolf, W. Meier, and M. Naya-Plasencia, "Conditional differential cryptanalysis of NLFSR-based cryptosystems," in Proc. ASIACRYPT 2010, Singapore, 2010, pp. 130–145.
[18] M. Ågren, M. Hell, T. Johansson, and W. Meier, "A new version of Grain-128 with authentication," in Proc. Symmetric Key Encryption Workshop, Lyngby, Denmark, Feb. 2011 [Online]. Available: http://skew2011.mat.dtu.dk/
[19] Banik, S., Maitra, S., Sarkar, S.: A differential fault attack on Grain-128a using MACs. SPACE 2012. LNCS, vol. 7644, pp. 111–125. Springer, Heidelberg (2012)
[20] Banik, S., Maitra, S., Sarkar, S.: A differential fault attack on the Grain family of stream ciphers CHES 2012. LNCS, vol. 7428, pp. 122–139. Springer, Heidelberg (2012).
[21] L. Ding and J. Guan, "Related Key Chosen IV Attack on Grain-128a Stream Cipher", IEEE Trans. Inform. Forensic Secur., vol. 8, no. 5, pp. 803-809, 2013.
[22] H. Zhang and X. Wang, "Cryptanalysis of stream cipher Grain family," Cryptology ePrint Archive, Report 2009/109, 2009, http://eprint.iacr.org/.
[23] Banik Subhadeep., Some Results on Sprout. INDOCRYPT 2015, LNCS 9462, pp. 124–139, 2015. DOI: 10.1007/978-3-319-26617-6 7
[24] A. Maximov, "Cryptanalysis of the "Grain" family of stream ciphers", in ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS'06), 2006, pp. 283–288.