

The Dancing Bear – A New Way of Composing Ciphers

Ross Anderson

Cambridge University

Abstract. This note presents a new way of composing cryptographic primitives which makes some novel combinations possible. For example, one can do threshold decryption using standard block ciphers, or using an arbitrary mix of different decryption algorithms – such as any three keys out of two AES keys, a 3DES key, an RSA key and a one-time pad. We also provide a new way to combine different types of primitive, such as encryption and signature. For example, Alice can construct a convertible signature that only Bob can verify, but which he can make world-verifiable using an AES key. We can incorporate even more exotic primitives, such as micropayments and puzzles, into compound constructs.

Previously, there had been two basic ways to combine cryptographic primitives. One could either design a compound primitive, perhaps using the homomorphic properties of discrete exponentiation, or one could embed several primitives into a protocol. Neither is ideal for all applications, and both have been extremely vulnerable to design errors. We provide a third construction that also allows the designer to do new things. We show, for example, how to incorporate cyclic dominance into a cryptographic mechanism, and how it might be used in a digital election scheme. Our new construction not only complements existing ways of composing crypto primitives; it also has the virtue of simplicity.

1 Introduction – Threshold Decryption

One of the great attractions of public key primitives such as factorization and discrete log is that we can often combine them into compound cryptographic operations such as threshold decryption, electronic cash and digital elections (see for example [22]). It is widely believed that public key cryptography is indispensable for such ‘fancy’ cryptographic schemes, as its homomorphic properties enable crypto primitives to be linked, blinded, and shared.

This belief is mistaken. We now show how to do threshold decryption using conventional crypto primitives such as AES and SHA, in a way that is not only practical, but inherits an established security proof. We will then go on to discuss the wider implications.

1.1 Background and notation

At FSE 93, Wheeler introduced WAKE, a variable-width block cipher based on an autokeyed table lookup [27]. This was aimed at giving a concrete implementation of the $\{X\}_K$ ‘curly-bracket’ abstraction for X encrypted by K that one finds in the protocol literature – without the sort of problems with modes of operation documented (for example) in [6]. At FSE 96, Anderson and Biham produced the BEAR construction, which also provides a variable-width block cipher but uses the composition of standard hash function and stream cipher primitives; one can thus construct (say) a 427-bit block cipher based on AES and SHA [4]. This is secure, though in the rather weak sense that a key-recovery attack on BEAR gives a key-recovery attack on the stream cipher and also finds either preimages or collisions for the hash function.

At FSE 97, Rivest remarked that such a cipher can be used with a fixed and indeed publicly-known key as an ‘all-or-nothing’ or ‘package’ transform. This was originally proposed as a means of pre-processing plaintext to ensure that an adversary performing a keysearch attack would have to decrypt all of a ciphertext before she could check whether her current key guess was correct or not [20].

At FSE 99, Jakobsson, Stern and Yung showed that if you take a message, add redundancy and an integrity check, then scramble it using a suitable package transform, then it suffices to encrypt just a single block of it. They proved that this construction is secure in the standard complexity model: indeed, it is secure against an adaptive chosen-message chosen-ciphertext attack [16]. They used the term ‘ideal length-preserving one-way function’ rather than ‘package transform’, but that is incorrect as the function cannot be one-way. It is better described as a single pseudorandom permutation of the desired length. However, there is no terminology for the combination of random nonce, message and integrity check, processed using such a pseudorandom permutation, so we’ll use the term ‘grizzle’: the grizzle of M under the nonce N is

$$G_N(M) = \Pi(N, M, c(N, M)) \tag{1}$$

where M is the input message, N is a random nonce, $c(N, M)$ is a checksum and Π is a fixed, publicly-known pseudorandom permutation for each length of input string. In a concrete implementation we might have $c(N, M) = \text{SHA}(N, M)$ and $P_i(x) = \text{BEAR}(0, x)$, that is, x BEAR-encrypted using the zero key.

We will need some more notation. We’ll write X_i for the i -th block of X , and assume that the block length b is clear from the context. Thus a Jakobsson-Stern-Yung encryption might be $\text{JSY}(K; M) = \{G_N(M)_1\}_K, G_N(M)_2, \dots, G_N(M)_k$: we grizzle the message M by the nonce N , encrypt the first block of it, and send that along with the other blocks which we do not encrypt. The Jakobsson-Stern-Yung result is that $\text{JSY}(K; M)$ is just as secure as a classical encryption $\{M\}_k$.

1.2 Basic construction

The first observation is that we can easily do shared decryption, in which n keyholders must each use their key to decrypt a message. Here, we grizzle the message as before and get each keyholder to encrypt a different block of it. Thus if keyholder i has key ki , the ciphertext is

$$c(k1, k2, \dots, kn; M) = \{G_N(M)_1\}_{k1}, \{G_N(M)_2\}_{k2}, \dots, \{G_N(M)_n\}_{kn}, \\ G_N(M)_{n+1}, \dots, G_N(M)_k \quad (2)$$

In other words, we encrypt the i -th block of the grizzled message $G_N(M)$ with the i -th key and transmit these blocks encrypted, together with the rest of the grizzled message unencrypted. (We assume here and in what follows that the message is long enough, that is, $k \geq n$.)

The Jakobsson-Stern-Yung proof goes across; from the viewpoint of each keyholder, the construction is secure against an adaptive chosen-message chosen-ciphertext attack, whether in the distinguishability or valid-pair-creation model. (In JSY terminology, valid-pair creation is trivially harder for more encryptions; the distinguishability proof implies that if there is one slave the attacker does not control then he cannot distinguish; it assumes nothing about the existence or otherwise of further slaves.)

It was always possible, of course, to perform shared decryption using secret-key algorithms if one used n stream ciphers, or with block ciphers if one were prepared to tolerate constraints on the order of decryption. So far, grizzling seems nothing revolutionary.

The second observation that we can get an $(n-1)$ -out-of- n threshold decryption mechanism by adding a parity block. With the above notation, we compute a simple xor-checksum of the blocks that will be encrypted:

$$C(M) = \oplus_{i=1}^n G_N(M)_i \quad (3)$$

and send it along with the message. As soon as $(n - 1)$ of the keyholders have decrypted their block, the remaining block may be trivially recovered by xor'ing $C(M)$ with the $(n - 1)$ plaintext blocks and the message can thus be decrypted. If fewer than $n - 1$ keyholders collude then no information about the plaintext can be recovered.

The third observation is the general case, that is, one in which any m out of n keyholders can decipher the message, but we do not wish any smaller subset of keyholders to obtain any information about the plaintext. To do this, we apply a suitable block-erasure error-correcting code to the grizzled plaintext. One may use an (n, m) Reed-Solomon code over an alphabet of size $q = 2^b$; such codes exist for $n \leq q$ [18]. A more sophisticated solution, which reduces the computational effort required for decoding with large n and m , is given by the recently-developed digital fountain codes of Byers, Luby and Mitzenmacher [7]. Yet another possibility is to use a secret-sharing scheme.

2 General Composition of Crypto Primitives

The above section showed how a combination of grizzling and block-erasure coding enables us to construct threshold-decryption schemes whose underlying crypto primitive is a standard block cipher such as AES. However, we nowhere assumed any property of this cipher, except that the encrypted blocks are the same length.

We can therefore use our construction with quite heterogeneous decryption algorithms, so long as we arrange to use equal block lengths. For example, we might use 2048-bit RSA for some shares, 16-block CBC-encryption with AES for others, and 32-block CBC-encryption with DES for still more. (The use of nonces in the grizzling construction makes IVs unnecessary.)

Heterogeneous keying of threshold primitives appears to be new, and may have uses in backward-compatibility applications. These are notoriously difficult to design in robust ways, and many attacks have been found that exploit backwards-compatibility features, in both protocols and APIs (see, for example, [5, 26]).

We'll now look at a few specific examples of new tricks.

2.1 Designated-confirmer and convertible signatures

As well as using different primitives that implement the same function, such as encryption, we can use our construction to compose different functions. Consider for example designated confirmer signatures [11], which allow one or more designated parties to confirm the authenticity of a digital signature without interacting with the signer. Such a construction is now almost trivial: Alice grizzles the message, signs one block of it, then encrypts another block in such a way that Bob can decrypt it. If $\sigma_{ka}(m)$ is a signature of m with message recovery, and $\{m\}_{kab}$ is an encryption with a key shared between Alice and Bob, then we might have

$$cs(M) = \sigma_{ka}(G_N(M)_1), \{G_N(M)_2\}_{kab}, G_N(M)_3, \dots, G_N(M)_k \quad (4)$$

It is now straightforward to arrange that the designated confirmer be any quorum of a group of keyholders, and, for that matter, that they can use any decryption scheme with which they are comfortable. This is a significant advantage of our new approach – functionality can be composed simply, compared with number-theoretic constructions where much work may be needed to combine two functions, and where many years can pass with schemes being proposed and then broken in the hunt for a new primitive (see for example comments in [15]). By making complex combinations simpler, our new construction makes them easier to analyze and understand.

3 Protocols

Number-theoretic constructions are not the only way to compose crypto. The other main technique is to use protocols. Ever-more complex functionality can be built up by successive rounds of interaction between two or more principals. However, interaction has its costs, even nowadays, and techniques for making one-pass versions of interactive protocols are generally limited to public-key mechanisms. Our new construction may allow more functionality can be packed into protocols that employ a single pass (or a small number of passes). We will give a practical example, from digital elections, in the next section.

Here, too, grizzling can help reduce the complexity that is the enemy of security. Crypto protocols are notoriously difficult to design correctly and understand completely. It is really important to have design methodologies that keep things as simple and comprehensible as possible [1, 3].

3.1 Example - digital elections

As an example of how grizzling might be applied in a real application, let us consider digital elections. At present, there are roughly two ways of doing online elections. There is a large literature on using public-key mechanisms to construct protocols that involve some measure of anonymity, such as digital cash and electronic elections; the literature draws heavily on seminal work by Chaum [8–10, 12], and for a recent survey see Rjašková [21]. These schemes typically return a single bit (‘Bush or Gore’). Then there are practical fielded systems that collect complex voting choices, in which the voter is invited to express multiple choices and may even be allowed to write in candidates of his choice. These systems are often flaky, and use at best ad-hoc security mechanisms [14].

A second issue is the human factor. Security usability may be the most neglected subdiscipline of security engineering, yet many existing e-voting proposals assume voters to be sufficiently computer-literate to install special client software on their machines. Our University is currently considering a move from exclusively paper-based ballots¹ to a system that will offer an electronic alternative. One of the strongly-felt requirements is that the new system should not place members of the School of Technology at an unfair advantage. This means that the system must be usable by professors of ancient languages whose computer skills are limited to operating a browser.

The proposed scheme, in its most basic form, is as follows. The voter receives, by physical mail, a voting ticket with a voter number N and an authenticator tag T . The voter number, in our University of 3,500 voters, could be a four-digit number. The tag is a random number too long to be brute-forced during the period of the election, and presented with sufficient redundancy to detect voter typing errors. (In practice, we might have a 72-bit tag with a 18-bit checksum presented as three blocks of five base-64 characters.)

¹ believe it or not, we vote in Latin – PLACET for yes and NON PLACET for no!

The voter completes an online ballot form with all the various voting options, which is presented by a signed applet. The applet then grizzles the ballot paper, asks the voter to enter his voting ticket, and validates the checksum. It then prints out a copy of the ballot paper, plus a substring of the grizzled version of the paper, to provide a voter-verifiable audit trail [14]. It then exclusive-ors the authenticator tag with the first 72 bits of the grizzled ballot paper, and returns to the web site the voter number N and the ballot paper which is now securely authenticated using the tag.

A surprising number of details have to be fixed to turn this into a properly engineered system. First, the anonymity can be provided by a mechanism such as Onion Routing [25] or Mixminion [13], but this is necessary anyway and the choice of such a mechanism is essentially orthogonal to electronic voting system design. (Simon pointed out in 1996 that an anonymous communications network is needed for anonymous payments, or the counterparty's identity can be determined simply by looking at the traffic data; and if such a network exists, then a much simpler digital cash system based on hash functions becomes possible [24]. Lee first applied this idea to digital elections in [17].)

Second, attention needs to be paid to separation of powers. There should be no single entity who can throw the election, miscount many votes, or break the anonymity of a large number of voters. Thus the anonymous remailers should be managed by more than one entity, while the voting tickets can be generated and mailed out using dual control mechanisms similar to those used to manage banking PINs. (This is not trivial but the details are beyond the scope of this paper; see [2] for more).

Third, we may require that voting be receipt-free, so that no-one can prove to a vote buyer how they voted. This is hard to square with voting at home. Indeed, there was a recent UK scandal in which a senior UK minister was accused of helping voters who were elderly, confused or easily intimidated to fill out postal ballot forms [19]. The best idea we've heard so far is that people should be able to vote as often as they like but only the last vote cast should count.

3.2 Scissors, paper, stone

Here is a novel suggestion. Each voter includes in their ballot one word of {scissors, paper, stone}. (For those not familiar with the childhood game, scissors cut paper, paper wraps stone, and stone blunts scissors – there is a cyclic dominance relationship.) If two ballots are received from the same voter, the returning officer will accept the dominant one. This way, even if the minister visits me just before the close of poll and persuade me to vote for her candidate, she has no better than a one in three chance if she chooses the token, and if I can choose it then I can make the vote invalid.

Schemes like these become possible with our ballot as it lets the election designer include arbitrary choices into the ballot paper. It is difficult to see how a classic digital election scheme could encode a cyclic dominance relationship.

Such cycles are of great importance in the theory of elections; see for example Sen [23]. (Freedom to design ballot papers is a double-edged sword; care need be taken lest the voter write into the ballot paper a recognition code given to him by a vote-buyer.)

It is not the purpose of this section to design an election system in detail. In fact, we still do not know how to design an election that meets all the theoretical and practical requirements that are thought reasonable. Our goal has been to show that grizzling a ballot paper, before authenticating it with a shared secret, may be an interesting new primitive for designers of real-world electronic election systems. It is striking, for example, that one can do authentication using xor.

4 Conclusion

The invention of public key cryptography caused great interest because it allowed us to do completely new things, and because it held out the promise of doing some existing key management tasks more efficiently. In this paper, we have shown that some of these new things – threshold decryption, designated confirmer signatures, and digital elections – can be done at least as easily by using an already-understood conventional cryptographic primitive, the pseudo-random permutation, in a new way. These three are just examples; we suspect that many public-key primitives of practical interest can be re-engineered using shared-key or heterogeneous mechanisms. We can also do some quite new things, such as encoding cyclic dominance relationships.

The more general importance of the new construction is that it provides a very general way of composing cryptographic primitives. Up till now, engineering a complex crypto function involved either using the homomorphic properties of public-key cryptography to design a custom primitive, or designing a protocol to bind together the required component functions. Both approaches are difficult; they can be fragile; and they have other limitations. Our new construction is comparatively simple and can often inherit existing security proofs. It may help designers reduce the dangerous complexity of existing approaches. It will not replace all protocols or all complex public-key primitives (for example, we do not yet know how to do threshold signature usefully). However, it promises to be a valuable addition to the security engineer’s toolbox.

Our work also suggests some further research problems. What, for example, is the best way to construct a fixed pseudorandom permutation of arbitrary length out of a given primitive, such as AES or SHA1? How should we re-engineer the existing primitives and protocols in the literature? What new primitives can we create, and what new applications now become possible?

Acknowledgements: David MacKay suggested the use of digital fountain codes and Eli Biham the use of secret sharing schemes. Three anonymous referees also provided useful comments on presentation.

References

1. M Abadi, RM Needham, “Prudent Engineering Practice for Cryptographic Protocols”, *IEEE Transactions on Software Engineering* v 22 no 1 (Jan 96) pp 6–15
2. RJ Anderson, ‘*Security Engineering – A Guide to Building Dependable Distributed Systems*’, Wiley (March 2001)
3. RJ Anderson, RM Needham, “Robustness principles for public key protocols”, in *Advances in Cryptology – Crypto 95*, Springer LNCS vol 963 pp 236–247
4. RJ Anderson, E Biham, “Two Practical and Provably Secure Block Ciphers: BEAR and LION”, in *Fast Software Encryption* (1996), Springer LNCS v 1039 pp 113–120
5. M Bond, RJ Anderson, ‘API-Level Attacks on Embedded Systems’, in *IEEE Computer* v 34 no 10 (October 2001) pp 67–75
6. C Boyd, WB Mao, “On a Limitation of the BAN Logic”, in *Advances in Cryptology – Eurocrypt 93*, Springer-Verlag LNCS v 765 pp 240–247
7. JW Byers, M Luby, M Mitzenmacher, “A Digital Fountain Approach to Asynchronous Reliable Multicast”, in *IEEE J-SAC, Special Issue on Network Support for Multicast Communication* v 20 no 8 (Oct 2002) pp. 1528 - 1540; earlier version as ICSI Technical Report TR-98-005, and SIGCOMM 1998
8. D Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms”, in *Communications of the ACM* vol 24 no 2 (Feb 1981) pp 84–88
9. D Chaum, “Blind Signatures for Untraceable Payments”, in *Proceedings of Crypto 82*, Plenum Press, New York, 1983, pp 199–203
10. D Chaum, “Security without identification: transaction systems to make big brother obsolete”, in *Communications of the ACM* vol 28 no 10 (Oct 1981) pp 1030–1044
11. D Chaum, “Designated confirmer signatures”, in *Advances in Cryptology – Eurocrypt 94*, Springer-Verlag (1994), pp 86–91
12. D Chaum, A Fiat, M Naor, “Untraceable electronic cash”, in *Advances in Cryptology – Crypto 88*, Springer LNCS v 403 pp 319–327
13. G Danezis, R Dingledine, N Mathewson, “Mixminion – Design of a Type III Anonymous Remailer Protocol”, in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*
14. D Dill, VerifiedVoting.org, <http://www.verifiedvoting.org/>
15. S Goldwasser, E Waisbard, “Transformation of Digital Signature Schemes into Designated Confirmer Signature Schemes”, at *First Theory of Cryptography Conference* (Feb 04) and MIT TR 329, March 2003
16. M Jakobsson JP Stern, M Yung, “Scramble all, encrypt small”, in ‘*Fast Software Encryption*’ (1999), Springer LNCS v 1636 pp 95–111
17. JH Lee, “The Big Brother Ballot”, in *Operating Systems Review* vol 33 no 3 (Aug 1999) pp 19–25
18. DJC MacKay, ‘*Information Theory, Inference and Learning Algorithms*’, Cambridge University Press, 2003
19. S Walters, D Turnbull, “Cabinet Minister in Vote Rigging Enquiry”, *Mail on Sunday*, May 4th 2003, pp 1, 8, 9
20. RL Rivest, “All-Or-Nothing Encryption and The Package Transform”, in *Fast Software Encryption* (1997), Springer LNCS v 1267 pp 210–218

21. Z Rjašková, 'Electronic Voting Schemes', at <http://people.ksp.sk/~zuzka/elevote.pdf>
22. B Schneier, 'Applied Cryptography', Wiley 95, ISBN 0-471-11709-9
23. A Sen, 'Collective Choice and Social Welfare', Holden-Day and Oliver and Boyd, 1970
24. DR Simon, "Anonymous Communications and Anonymous Cash" in *Advances in Cryptology – Crypto 96*, Springer LNCS v 1109 pp 61–73
25. PF Syverson, DM Goldschlag, MG Reed, "Hiding Routing Information", in *Information Hiding 1996*, Springer LNCS v 1174 pp 137–150
26. D Wagner, B Schneier, "Analysis of the SSL 3.0 Protocol", in *The Second USENIX Workshop on Electronic Commerce*, Proceedings, USENIX Press, November 1996, pp 29–40
27. D Wheeler, "A Bulk Data Encryption Algorithm", in *Fast Software Encryption 1993*, Springer LNCS v 809 pp 127–134