Big Brother Is Watching You: A Closer Look at Backdoor Construction

Anubhab Baksi[1(✉)], Arghya Bhattacharjee[2], Jakub Breier[3], Takanori Isobe[4], and Mridul Nandi[2]

[1] Nanyang Technological University, Singapore, Singapore
anubhab001@e.ntu.edu.sg
[2] Indian Statistical Institute, Kolkata, India
[3] Silicon Austria Labs, Graz, Austria
jbreier@jbreier.com
[4] University of Hyogo, Kobe, Japan
takanori.isobe@ai.u-hyogo.ac.jp

Abstract. With the advent of Malicious (Peyrin and Wang, Crypto'20), the question of a cipher with an intentional weakness which is only known to its designer has gained its momentum. In their work, the authors discuss how an otherwise secure cipher can be broken by its designer with the help of a secret backdoor (which is not known to the user/attacker). The contribution of Malicious is to propose a cipher-level construction with a backdoor, where it is computationally infeasible to retrieve the backdoor entry despite knowing how the mechanism works.

In this work, we revisit the work done by Peyrin and Wang in a greater depth. We discuss the relevant aspects with more clarity, thereby addressing some of the important issues connected to a backdoor construction. The main contribution, however, comes as a new proof-of-concept block cipher with an innate backdoor, named ZUGZWANG. Unlike Malicious, which needs new/experimental concepts like partially non-linear layer; our cipher entirely relies on concepts which are well-established for decades (such as, using a one-way function as a Feistel cipher's state-update), and also offers several advantages over Malicious (easy to visualise, succeeds with probability 1, and so on). Having known the secret backdoor entry, one can recover the secret key with only 1 plaintext query to our cipher; but it is secure otherwise.

Keywords: Backdoor · Hash function · XOF · Block cipher · Feistel · Low-MC · Malicious · Low-MC-M · Provable security · SPRP · White-box

1 Introduction

One of the problems that comes with designing a cipher is to gain the collective trust of the community. The cipher must satisfy certain security requirement with sufficient margin to prevent a malicious attacker (who has the full knowledge of

L. Batina et al. (Eds.): SPACE 2022, LNCS 13783, pp. 81–96, 2022.
https://doi.org/10.1007/978-3-031-22829-2_5

the cipher specification) from getting information secured by the cipher under a secret key. At the same time, it is also essential that the cipher designer will fail to retrieve the data secured by the cipher under a secret key. Stated in other words, the designers of a cipher have to convince the rest of the community that the cipher does not have a hidden vulnerability that evades known cryptanalytic methods (thus, it is known only to the designers). As we have seen, this is not always the situation. Case in point, it has long been speculated that the SIMON and SPECK [4] family of block ciphers have some form of hidden backdoor (see [9] or Schneier's blog[1]. among other sources[2]), which are only known to the designers[3]. Despite years of speculation, the presence of any backdoor is not determined.

Amidst such situation, it is not surprising that the cryptographic community will take interest in the prospect of designing a cipher with an implanted backdoor. We have recently seen this happening in the Crypto'20 paper [9] where the designers take an otherwise secure cipher family and implant a backdoor in it. They present their contribution in the form of a framework, named, Malicious. It works by querying the cipher with a chosen tweak difference on a variant of the LOWMC [1] family of ciphers (this tweak difference is secret and known only by the cipher designer). Ultimately, this tweak difference propagates through the cipher in such a way that the resulting ciphertext difference allows the cipher designer to retrieve the secret key (the secret key is chosen by, and only known to the user) with a certain probability. They also describe a Malicious based tweakable block cipher, named LOWMC-M.

1.1 Contribution

A big part of the inspiration of our work goes to the Crypto'20 paper by Peyrin and Wang [9]. More precisely, we take a deeper look at the Malicious framework (and its instance LOWMC-M), and improve the state-of-the-art in a number of ways.

To begin with, we show a provably secure construction of backdoor that improves from LOWMC-M [9]. Our method of the backdoor construction relies entirely on pre-existing notions of security, which are well-known/well-analysed for decades. The construction of Malicious is more on the experimental side, that relies on lesser studied concepts such as partially non-linear layer. Apart from that, our backdoor requires only 1 plaintext query (works with probability 1), unlike the LOWMC-M that requires a number of chosen (plaintext, tweak) queries. We do not need any tweak, and the overall idea is generic – it can be implemented atop virtually any encryption and hash algorithm[4]. Thus, making it possible to have a backdoor without any tweak and not tied to LOWMC [1].

[1] https://www.schneier.com/blog/archives/2018/04/two_nsa_algorit.html.

[2] It is also worth pointing out that problem is partly exacerbated due to the absence of any cryptanalytic result in the introducing paper [4].

[3] In this case, the designers are a group of researchers from the American government's National Security Agency (NSA), possibly hinting at a government-level initiative in the background.

[4] Depending on the hash output size and the state size of the encryption algorithm, we may need to pad/truncate.

The coverage/contribution of our paper does not end there. We ask several relevant questions, which have not been answered yet. We argue that no matter how cleverly the backdoor is designed, it is not possible for the designer to access it without the user's cooperation (as the user can always cross-check if some secret information is revealed – and if so – can deny the request); or one backdoor entry cannot be used more than once (as the attacker will get to know as soon as it is used). The elephant in the room, however, lurks in hiding the key which is released as a result of the backdoor access—the key is not encrypted in any way, meaning the attacker gets to know about it no later than the designer does.

1.2 Prerequisite

As discussed in [9, Section 1], the concept of backdoor itself is not new. In our context, we directly follow [9]. For clarity, the terms/ideas used are briefly described here.

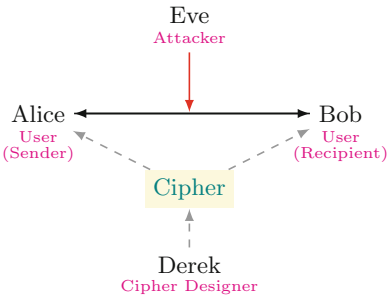

Fig. 1. Schematic of backdoor work-flow

The cipher designer, whom we refer to as *Derek* for simplicity, designs a cipher with an intentional backdoor (which is known only by him). The cipher (the public description of the cipher, to be more precise) is then used by the users, Alice and Bob, to communicate sensitive information. The attacker, Eve, watches the channel between Alice and Bob closely and knows all the (publicly available) specification/cryptanalysis regarding the cipher. Figure 1 shows a schematic representation.

Now, at some point during communication, Derek can use the backdoor to retrieve sensitive information, this incidence (if happens) is indicated by backdoor *access*. The backdoor *mechanism* lets Derek to access sensitive information (this works as a weakened version of the cipher). The mechanism is activated with the help of a backdoor *entry* (e.g., a 128-bit string when used as the plaintext to the cipher), which is known only to Derek (it will likely become a public knowledge after it has been used once).

For the interest of brevity, we assume the reader's familiarity with the basic terms/concepts, including; CSPRNG, LFSR, block cipher (along with padding, and mode of operation like `CTR`), stream cipher (along with IV and nonce), hash function, MAC, AE, AEAD; PRP, SPRP; OWF ; cipher families (Feistel, SPN and ARX); and ciphers (`DES`, `AES`, `RC4`, `RSA`). We also use XOF[5] (eXtended Output Function). An XOF is a one-way function that takes a message of arbitrary length and returns a message of desired length (i.e., $\{0,1\}^\star \rightarrow \{0,1\}^\star$).

1.3 Organisation

The background information is covered in Sect. 2 (particularly Sect. 2.2 contains some previously unreported observations). Section 3.1 goes through the practical aspects of a backdoor, and Sect. 3.2 covers two related notions of security.

In Sect. 4, we present our block cipher named "`ZUGZWANG`"[6] that has a backdoor[7]. After the fundamental idea is stated in Sect. 4.1, we show a concrete instance by using `AES-128` and `SHAKE-128`[5] in Sect. 4.2. Apart from that, a comparison with `Malicious` is given in Sect. 4.3.

The conclusion can be found in Sect. 5. For more discussion (along with security proof and test cases) one may refer to the extended version available at [3].

2 Background

2.1 Implementation Level and Cipher Level Backdoors

The term, 'backdoor' is generally more common in the cyber-security or hacking communities. Here it typically refers to an intentionally implanted weakness[8]. This process is done at the fabrication/implementation level and transparent at the cipher design level[9]. See [10] for a recent example.

Our interest, however, lies on the other type of backdoor, which works at the cipher design level. In this case, the cipher is so designed that, it has a secret backdoor which is known only by its designers. It is long been speculated that some ciphers, whose entire specification is available in public, may contain some secret backdoor.

[5] See https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf.

[6] It is a German word (translates to 'a compulsion to move'), used in context of Chess to describe wherein all the available moves for a player make the situation worse.

[7] As it has a backdoor, any practical application of `ZUGZWANG` is not recommended (to be used mostly, if not only, as an interesting proof-of-concept).

[8] For instance, one may look at the "politically correct" backdoor: https://www.kb.cert.org/vuls/id/247371.

[9] This is noted in [9, Section 1]: "There are two categories of backdoors. The first one is the backdoor implemented in a security product at the protocol or key-management level, which is generally considered in practice.".

2.2 Context

The cipher level backdoors can be theoretically divided into the following categories:

(1) A cipher so craftily designed that nobody is able to find the presence of a backdoor after few years of speculation and testing. So far it passes all the known methods of cryptanalysis. It is not known whether there is actually a backdoor or this is just a myth. If/when this cipher is made a standard and adopted as a global standard, it is in theory possible for its designers to access the backdoor and retrieve sensitive information.
The `SIMON` and `SPECK` [4] family contain few block ciphers which are suspected to have this kind of backdoor (see, e.g., [9]). It is not known if there is any backdoor, or how the backdoor mechanism works; if there is any in the first place. If there is some backdoor in those ciphers, it is never accessed, to the best of our knowledge/understanding.

(2) A cipher where the designers publicly claim there is a backdoor. The cipher is secure except when the backdoor is accessed. The designers make no attempt to hide the backdoor; rather they claim upfront that there is a backdoor – this is how the backdoor mechanism works – so on and so forth.
This category is recently popularised through the `Malicious` framework [9]. This framework can create such a cipher by tinkering with some otherwise secure cipher, given the base cipher satisfies some criteria. By accessing the backdoor, the cipher designer can retrieve the key by analysing the cipher output. The `LOWMC-M` [9] is an instance of this framework, which takes a secure instance of `LOWMC` as the base cipher.
One may notice the following characterisation of this category:
   (α) The presence of the backdoor is made public by the designers. This also nullifies the question of whether it is hard to spot the presence of the backdoor had it not been known[10].
   (β) Except when the backdoor is accessed, the cipher is secure. When the backdoor is accessed, the secret key is released from the cipher output (assuming the user does not prevent that, see Inference (B)) – at least in theory, satisfying the "practicality" condition of [9, Section 2.2].
   (γ) Though the backdoor mechanism is public, it is infeasible to find out what the secret backdoor entry that activates the mechanism is. This is called the "undiscoverability" condition [9, Section 2.2].

By observing the Category (2) design in the literature (namely, [9]), the following inferences can be drawn (see Sect. 3.1 for more discussion):

(A) The backdoor can be accessed at most once. The backdoor mechanism is public, therefore anyone can check the incoming requests to the user to see if the backdoor is activated. Once it is found, the secret backdoor access

[10] It may be hard to spot the backdoor for someone who does not know beforehand, but here it does not matter as the designers have already made it public.

becomes visible to everyone. Basically, one can monitor all the incoming traffic to the user and attempt to reverse-engineer the backdoor entry, and will eventually succeed as soon as the backdoor entry is used.
This can be done by the user (Alice) or the attacker (Eve[11]) alike. Though Eve does not know the actual secret key (which is chosen and kept secret by Alice), she can still choose an arbitrary key, and then follow-through the steps of the backdoor mechanism to see if information about the arbitrarily chosen key is released.

(B) It is not possible to extract the secret key without the user's (Alice's) cooperation. Alice can always keep an eye out for activation of the backdoor mechanism. Based on that, she may return an invalid output or something random (if the backdoor mechanism is activated), instead of the actual output from the cipher.

(C) The key which is released from the cipher output (as a consequence of the backdoor access) is not encrypted[12], meaning pretty much everyone on the network (including the attacker) can access it.

In this work, we aim at improving Category (2) backdoors; i.e., we are interested to create improved design that satisfies Criteria (2α), (2β) and (2γ). It is important to note that those criteria are adopted from [9], and are not conceived by us. It is perhaps worth noting that Inferences (A) and (B) violate the "untraceability" condition which is described in [9, Section 2.2] (Inference (B) is already acknowledged in [9, Section 5.3] as a violation of "untraceability"). Whether or not it is possible to design a Category (1) backdoor is left as an open problem.

Remark 1. The closest to Category (1) the designers of `Malicious` could have gone is to present a new cipher/framework/mode and make a vague claim about presence/absence of a backdoor. Then it would be up to the community to figure out if there is a backdoor, how to activate the backdoor/how the backdoor mechanism works, etc.

Remark 2. In theory, it is possible to design a cipher in Category (1) if the designer manages to find an attack not yet known to the (mainstream) community[13]. The backdoor in this case will be activated through this new attack, and will (more than likely) be missed by the community (at least until this attack

[11] As per [9, Section 2.1], the attacker/eavesdropper Eve is considered within the `Malicious` framework.

[12] If the released key is encrypted with another key, that means the cipher designer and the user have to know the other key beforehand. In that case, they can simply use any cipher (with the other key) to communicate the key released through the backdoor instead, thus completely cutting off the need for a backdoor.

[13] For instance, some of the public-key ciphers (including `RSA`) are now known to be vulnerable against quantum computers, but those attacks were not known when those ciphers were designed. In a less restricted sense, the quantum attacks can be considered as backdoors to those ciphers.

is discovered or the backdoor mechanism is reverse-engineered). For perspective, the construction of `Malicious` [9] depends on the well-studied differential attack; thus the backdoor, in a very high likelihood, would be spotted by the community (had it not been known already).

3 Basic Concepts

3.1 Practical Application of a Backdoor

Status Quo. The first problem that arises while talking about the practicality of backdoor is to convince the users to adopt it. There is no shortage of efficient ciphers in the public domain; with well-described design rationale and which are well-analysed by the community. The users, Alice and Bob, may simply refuse to adopt any new/experimental cipher (for example, any cipher from the `LOWMC` family [1] altogether, or the unusual choice of using an XOF to design an encryption as in `LOWMC-M` [9]), suspecting there could potentially be a backdoor. Therefore, in a loose sense, they agree for the designer to retrieve the secret key if they agree to adopt a new cipher. Thus, the design and study of backdoor appears to be purely an academic interest than a pragmatic one.

Anyway, as far the technical problems are concerned with the current concept of backdoor [9] (which we call Category (2), see Sect. 2.2), we note the following: Since the identity of the cipher designer (whom we call Derek for simplicity, as indicated in Sect. 1.2) is known to everybody in the network; including Alice (sender), Bob (recipient) and Eve (attacker). Therefore Alice (as well as Bob) can be extra cautious when a request comes from Derek, implying the limitations:

(i) Alice can simply deny any request from Derek, preventing him to access the backdoor.
(ii) If Alice complies with Derek's requests and lets him access the backdoor, this can be noticed by Eve. Now the secret key is leaked through the response from Alice and the key is not encrypted[14], thus Eve can effectively recover the key.

Overall, the Limitations (i) and (ii) mostly, if not fully, diminish any real-life application for a Category (2) backdoor. The cipher designer (Derek) cannot use it without active cooperation from the user (Alice or Bob). Even if Derek can obtain anonymous identity or spoof a fake identity, it is still up to the mercy of Alice. All the information is coming from Alice, so she can simply check the output from the cipher before sending it[15]; and discard the request or give a random output; should she suspect the backdoor is being accessed. On

[14] There is practically no way to encrypt this key, at least within the realm of symmetric-key cryptography; as this would require exchange of another secret key between Alice and Derek. This invalidates the need for a backdoor in the first place.

[15] For instance, Alice can check if the XOR of two consecutive cipher outputs equals to the key. Given the backdoor mechanism is public, she already knows exactly what to look out for.

the other hand, if Alice agrees Derek to access the backdoor, they can instead create a secure channel between them (no need for a backdoor). Besides, letting Eve know the secret key is a miserable flaw, since the whole purpose of any cryptographic system is to ensure the attacker cannot access the key.

The point to note here is, we are heavily implying that the notion of backdoor, at least in its current form, suffers from severe limitation that comes from lack/absence of trust for Derek. If Alice does not respond to anyone she does not trust, anonymous/fake identity by Derek is meaningless. We are not saying either of the assumptions is objectively true/untrue. We are simply saying, in order for Derek to succeed in utilising the backdoor; he needs to circumvent those real-life problems at first, which may turn out to be challenging.

Uncertain Future Prospect. While it does not seem possible to extract the secret key without cooperation from the user, it may be possible with some cipher in the future where the designer can extract the key in a way that the attacker cannot get it. One potential concept to achieve this in the future (that may or may not work) can be stated as follows.

Suppose, instead of only one backdoor entry, it is split into q backdoor shares[16] (somewhat comparable with the concept of secret-sharing [11]), where the cipher output from all the shares are required to retrieve the key. Say, by querying with the b_i backdoor entry, c_i is obtained, for $i = 0, \ldots, q-1$. Each c_i contains some information about the secret key, but all of those are required to get the key.

Not only that, each c_i is connected in secret way (which is only known to Derek) so that the connection is to be respected in order to find the key. With some suspension of disbelief, say, $k = \mathbf{f}(c_{j_0}, c_{j_1}, \ldots, c_{j_{q-1}})$ where the function $\mathbf{f}$ is secret (only known to Derek) and is not symmetric, for $(j_0, j_1, \ldots, j_{q-1})$ being secret a permutation of $(0, 1, \ldots, q-1)$. Thus, despite knowing all the public information as Derek does, Eve may not be able to actually uncover the key given certain regularity assumptions (like, q is sufficiently large) as she would need to cover the search-space of $q!$.

This concept is shared here only to pique the interest of the future researchers. Whether or not this will turn out to be a feasibility is unclear as of now.

3.2 Associated Notions of Security

Undetectability. The authors of `Malicious` in of [9, Section 2.2] mention one desirable security notion for a Category (2) backdoor, "undetectability". It is defined as *"the inability for an external entity to realize the existence of the hidden backdoor"*. Here we argue that this is a bit tricky.

Note from Criterion (2α), the backdoor designers of `Malicious` [9] have already made the presence of the backdoor a public knowledge. Thus, it is a pre-conceived knowledge that a backdoor exists, thus violating the "undetectability".

[16] Possibly something similar is laid out by Peyrin: https://thomaspeyrin.github.io/web/assets/docs/invited/TII_CRC_21_slides.pdf, Slide 63.

We further notice that the notion about whether the cipher has an embedded backdoor does not seem to hold either. This is because we are not aware of any possible way to ascertain a cipher does not contain a backdoor (*"How do you know* `AES` *does not have a backdoor?"*). The ciphers which are broken can be (arguably) considered to have a backdoor, but it does not seem possible to comment on non-existence of a backdoor about those ciphers which are deemed secure. As a consequence, it is not possible to say an arbitrary instance of `LOWMC-M` does not contain a backdoor (regardless of an intentional backdoor following `Malicious` is implanted or not).

Need for White-box Security. Given the analysis in Sects. 2.2 and 3.1; it stands to reason that, Alice (as well as Bob) and Eve can reverse-engineer the backdoor mechanism as soon as the first query is made by Derek as long as the cipher specification is public. Indeed, no matter how the backdoor mechanism works, it has to trigger something (such as, some variable has to become 0, some loop has to terminate, and so on). If the cipher specification is known, then anybody can utilise such information no later than the correct backdoor entry is used.

Therefore, if we want the backdoor mechanism will not be revealed even after a backdoor entry is queried with, a basic condition is that the cipher specification is to be kept secret by Derek. However, this alone is not enough, since it is possible to reverse-engineer the cipher specification given its (unprotected) implementation (cf. the well-known cases of reverse-engineering `RC4`[17] or `CRYPTO-1` in Mifare Classic RFID tag [8]). Thus, the implementations of the cipher (which are prepared and shared by Derek to Alice and Bob) practically have to be secure against the *white-box* [5,6] attacks. In a white-box setting, the secret key is embedded in the cipher implementation in a way that it cannot be recovered. That said, one may notice the following differences from the usual white-box setting (cf. *obfuscation*[18]):

1. The cipher specification itself is secret in a backdoor setting, which is a more stringent requirement than usual white-box (where it is public).
2. The cipher designer supplies the implementations to the users, but he does not know the secret key. This contrasts the usual white-box setting where the implementer knows and embeds the key. It is not clear whether this is a more stringent requirement.

At this point, it is perhaps safe to assume, there is no proper real-life application of the concept of backdoor introduced in [9], at least in the mainstream academic community. Somebody may still use a cipher like that if it is enforced[19].

[17] https://web.archive.org/web/20010722163902/http://cypherpunks.venona.com/date/1994/09/msg00304.html.

[18] https://www.esat.kuleuven.be/cosic/blog/program-obfuscation/.

[19] One may compare with the government-issued (closed-source) applications to trace COVID-19 to some extent, though there is no separate recipient (Derek = Bob) and there is no secret key to recover.

For instance, assume the situation where there a push from the government to implant some intentional backdoor to compromise the security of products used by the common people. In that case, it is in theory possible to use a Category (2) cipher, with its full specification being available in public (and with no white-box protection). Our cipher ZUGZWANG can be in theory used in such a situation; but as academic researchers with a moral compass, we do not condone that. To the best of our finding, the only incident similar to this is rumoured in Australia back in 2018, but it seems to be officially denied[20].

4 ZUGZWANG: Constructing a Block Cipher with a Backdoor

One major observation from Malicious [9] is that, the only reason the user/attacker cannot retrieve the backdoor is the one-way property of the XOF. As it is known, a Feistel block cipher can use an OWF as its state-update (see, DES for an example), we adopt the idea to finally extend it to ZUGZWANG. Whether or not a similar construction is possible with SPN and ARX families, and whether some other idea is possible that does not involve any OWF, are left open for future research.

4.1 Fundamental Idea of ZUGZWANG

In its simplest form, ZUGZWANG is a 2-branch balanced Feistel network based block cipher that runs for n rounds (counting of rounds goes from 0 to $n-1$). It uses $f_i(K_i, c_L)$ as the round function for the i^{th} round; where K_i is the corresponding round key, c_L is the plaintext or the intermediate ciphertext currently at the left branch. Each f_i has the property that it collapses if $c_L = \hat{p_0}$ (if i is even) or $c_L = \hat{p_1}$ (if i is odd), for some predefined $\hat{p_0}$ and $\hat{p_1}$. In this case, $\hat{p} = \hat{p_0}||\hat{p_1}$ constitutes the secret backdoor. Also note that, the last Feistel round does not have any swap operation between the two branches (so there are $n-1$ branch swaps).

Now, notice that, $\hat{p_0}$ and $\hat{p_1}$ cannot be used directly in the specification of f_i's (those cannot be passed as parameters of f_i's); otherwise Alice and Eve would trivially retrieve these. Thus, we run an OWF, $H(\cdot)$ first. This leads to pre-computing and storing $H(\hat{p_0})$ (respectively, $H(\hat{p_1})$) where i is even (respectively, odd) in the cipher specification as constants. Now that $H(\cdot)$ is used to $\hat{p_0}$ and $\hat{p_1}$, we need to apply it to c_L too. Ultimately, instead of directly checking whether $c_L = \hat{p_0}$ or $c_L = \hat{p_1}$, we are now checking if $H(c_L) = H(\hat{p_0})$ or $H(c_L) = H(\hat{p_1})$.

Although $H(\hat{p_0})$ and $H(\hat{p_1})$ are stored and accessible in the cipher specification, the following claims hold due to the property of $H(\cdot)$:

- **Pre-image resistance:** It is hard to retrieve $\hat{p_0}$ and $\hat{p_1}$.
- **Second pre-image resistance:** It is hard to find another $\hat{p_i}'$ ($\neq \hat{p_i}$) such that $H(\hat{p_i}') = H(\hat{p_i})$ for $i = 0, 1$.

[20] https://www.homeaffairs.gov.au/about-us/our-portfolios/national-security/lawful-access-telecommunications/myths-assistance-access-act.

Notice that the birthday-bound security for collision does not apply here. These claims constitute the secrecy of the backdoor entry given its H; i.e., no algorithm better than the brute-force search is known.

Next, we need to use an encryption, $E_{K_i}(\cdot)$ inside f_i's to make the overall Feistel cipher secure. As some form of the check $H(c_L) = H(\hat{p_j})$ for $j = i \pmod 2$ has to be inside f_i, a natural choice is $f_i = E_{K_i}(H(c_L)) \oplus \alpha_i$ where $\alpha_i = E_{K_i}(H(\hat{p_j}))$. After n rounds of update are done, post-whitening keys (K_L, K_R) are XORed to the branches.

Notice that when $\hat{p_0}$ and $\hat{p_1}$ are fed to the left and right branches respectively, f_i's are always 0, and this holds irrespective of number of rounds. Basically, f_i (which is 0) is XORed to the right branch, making the content of the right branch going to the left branch in the next round. As the content of the right branch is what makes f_{i+1} collapse, the pattern continues. This allows us to create an arbitrary number of rounds, though for proper security in two directions (and with the assumption that E and H have desirable cryptographic properties) the number of rounds is required to be ≥ 4. Therefore, when fed with the secret backdoor $\hat{p} = \hat{p_0}||\hat{p_1}$, the ciphertext from the cipher does not depend on the Feistel round keys (K_i's). Rather it always follows the simple invariant relationship with the post-whitening keys (K_L, K_R), which leads to direct key recovery with 1 plaintext query as:

$$(c_L, c_R) = \begin{cases} (\hat{p_0} \oplus K_L, \hat{p_1} \oplus K_R) & \text{if number of Feistel rounds is odd,} \\ (\hat{p_1} \oplus K_L, \hat{p_0} \oplus K_R) & \text{if number of Feistel rounds is even.} \end{cases}$$

On the other hand, when a p $(\neq \hat{p})$ is used as the plaintext, the cipher works as secure Feistel block cipher. At each round, the state update can be compared to a Boolean derivative of E – it resembles a form of differential attack on E (but weaker since $H(\hat{p_0})$ and $H(\hat{p_0})$ constants). Given E is secure, any differential attack on E does not give any usable information. We thus conclude, 4 rounds of the `ZUGZWANG` construction can be considered to provide adequate security in two directions.

Extension to Other Symmetric-key Primitives. It may be possible to extend the core idea of `ZUGZWANG` to other primitives in the symmetric-key cryptography (viz., stream cipher, hash function, MAC, AE and AEAD). However, it is not immediately apparent how such extension will pan out. For instance, it is possible to get a stream cipher from a block cipher by using a number of modes (e.g., `CTR`); but it is to be noted that the plaintext does not enter the state of a stream cipher per-se. As such, we may have to use a secret IV/nonce so that, (say) the key-stream becomes all-zero regardless of the secret key. This requires elaborate discussion, and hence is kept out-of-scope for this work.

Feistel Types, Branches and Rounds. The basic idea can be generalised to more Feistel branches, wherein the secret backdoor entry is split into multiple

branches. In that case, one also needs to decide on the type of the Feistel Network and the minimum number of rounds required. Analysis such such options is left open for future research.

Table 1. Complexity of whitening key recovery in 128-bit ZUGZWANG construction

(a) 2-branch Feistel

Whitening		Backdoor Complexity	
#Pre	#Post	Encryption	Decryption
$0^{\ddagger}$	$2^{\ddagger}$	2^0	2^{128}
1	1	2^{64}	2^{64}
2	0	2^{128}	2^0
2	2	2^{128}	2^{128}

‡: Instantiated in Sect. 4.2

(b) 4-branch Feistel

Whitening		Backdoor Complexity	
#Pre	#Post	Encryption	Decryption
0	4	2^0	2^{128}
1	3	2^{32}	2^{96}
2	2	2^{64}	2^{64}
3	1	2^{96}	2^{32}
4	0	2^{128}	2^0
4	4	2^{128}	2^{128}

Location of Whitening Keys/Backdoor on Decryption. Note that, the key recovery through backdoor access in ZUGZWANG does not retrieve any Feistel round key, rather it retrieves the whitening keys (the post-whitening keys for encryption, to be more precise). If we take all the Feistel branches have a whitening key XOR (for maximum key recovery), then the question is whether to use pre- or post-whitening keys.

For simplification of notation, assume that we have a 128-bit and 2-branch ZUGZWANG construction with n-rounds. First, let us study the situation for encryption where the left branch has a pre-whitening key (K'), the right branch does not have a pre-whitening key. In this case, $\hat{p_0}$ does not make the f_i's collapse for even rounds, but $\hat{p_0} \oplus K'$ does. Since the designer does not know which K', he has to brute-force over 64-bits. Therefore, he has to query with $\hat{p_0} \oplus K' || \hat{p_1}$ for all possible 2^{64} choices of K'. The correct guess of K' can be identified by the output at the end (which will depend on presence/absence of whitening keys and if n is even/odd).

Reflecting on this, we observe that the construction with post-whitening keys helps in backdoor access in encryption, but not in decryption. Similarly, pre-whitening helps in decryption, but not in encryption. The invariant property that the product of the complexities for the whitening key recovery at both sides remains the same as the brute-force search. This is an inherent property of the ZUGZWANG construction. As shown in Table 1, this cannot be improved by increasing the number of Feistel branches. Improving the whitening key recovery complexity from two sides can be considered as a future work. Further, as indicated in Table 1, if both the pre- and post-whitening keys are used; this particular backdoor mechanism ceases to exist.

To the best of our finding, no claim about the backdoor access from Bob's side (i.e., decryption) is available in Malicious [9]. Thus, the notion of "practicability", which is introduced in [9, Section 2.2], is unclear for Malicious/LOWMC-M decryption.

4.2 A Concrete Instance of ZUGZWANG (Using AES and SHAKE)

We show an instance of ZUGZWANG[21] that uses a 2-branch balanced Feistel structure, 128-bit state, runs for 4 Feistel rounds, and uses 0 pre-whitening and 2 post-whitening keys. We choose AES-128 for encryption (E), and SHAKE-128 as XOF (H). See Algorithm 1 for its formal description.

The basic construction for ZUGZWANG is as shown in Fig. 2. The 128-bit master key K and the 128-bit backdoor entry $\hat{p}$ are split into two 64-bit post-whitening keys: $K = K_4||K_5$, $\hat{p} = \hat{p_0}||\hat{p_1}$ respectively. The 128-bit Feistel round keys (K_i for $i = 0, 1, 2, 3$) are generated by running AES in CTR mode with key k (i.e., with i as the plaintext).

As per the construction, the $H(\cdot)$ of $\hat{p_0}$ and $\hat{p_1}$ are computed and stored. Since these are to be used as the plaintext for AES-128 (Line 10), we take 128-bit output for these. Similarly, the plaintext/intermediate ciphertexts are to be used in AES as plaintexts (Line 11), so the outputs from SHAKE for these are also taken as 128 bits long. However, since each Feistel branch is 64 bits long (see Line 12), we truncate the last 64 bits of these 128-bit SHAKE outputs.

If $\hat{p}$ is not known, then we claim this concrete instance offers 128-bit security. On the other hand, if queried with $\hat{p}$ as the plaintext, then the post-whitening keys (K_4, K_5) are revealed.

ZUGZWANG is not meant to be used in practice, rather its primary function is to work as a proof-of-concept. Thus, we acknowledge the device footprint for the concrete instance can be significantly lowered (say, using less number of rounds for AES, replacing AES with a lightweight encryption, using an LFSR to generate Feistel round keys, etc.) but do not make any attempt to do so. For the same reason, we do not present any benchmark.

4.3 Comparison of ZUGZWANG with Malicious/LOWMC-M

In essence, the fundamental concept in Malicious [9] can be as described as follows. The backdoor entry is accessed through a (secret) difference at the (public) tweak and the (public) plaintext. The backdoor access works by cancelling the differences with one another in such a way that ultimately there is a high probability differential trail at the end, which potentially leaks the key. However, if just this much would be implemented, the attacker/user would (likely) notice the differences applied through the tweak and the plaintext, which would in turn reveal the backdoor entry. To prevent the attacker/user from obtaining the differences, the designers of Malicious [9] pass the differences through an XOF.

In essence, the fundamental concept in Malicious [9] can be as described as follows. The backdoor entry is accessed through a (secret) difference at the (public) tweak and the (public) plaintext. The backdoor access works by cancelling the differences with one another in such a way that ultimately there is a high probability differential trail at the end, which potentially leaks the key. However, if just this much would be implemented, the attacker/user would (likely)

[21] We use the same term, 'ZUGZWANG', to indicate the overall construction idea as well as the concrete instance.

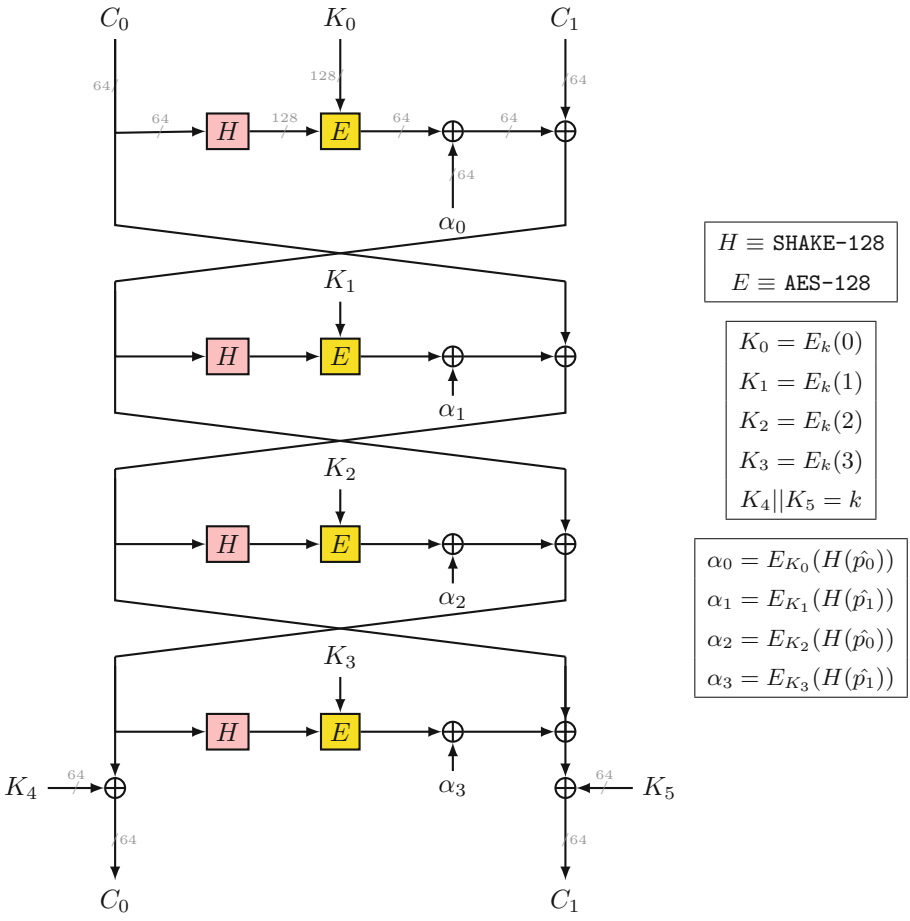

Fig. 2. ZUGZWANG (concrete instance/encryption)

notice the differences applied through the tweak and the plaintext, which would in turn reveal the backdoor entry. To prevent the attacker/user from obtaining the differences, the designers of Malicious [9] pass the differences through an XOF.

The following comparative points can be noted:

1. Malicious is based on relatively novel and not so much analysed design principles. It cannot be implemented atop of any pre-existing cipher. Effectively, Malicious is reliant on the security of LOWMC [1]. Besides, being a new design type, itself requires its own analysis (in particular, LOWMC-M is revised in [9, Section 4.3] after a new analysis is presented in [7] where 7 instances of LOWMC-M with original parameters are broken without finding backdoor by algebraic attacks on LOWMC). ZUGZWANG can be designed using already

Algorithm 1: `ZUGZWANG` – concrete instance/encryption (128 bits, 2-branch Feistel, 4 rounds; `AES-128` as encryption, `SHAKE-128` as XOF)

Input: $\hat{p}$ (backdoor entry, 128 bits), p (plaintext, 128 bits), k (master key, 128 bits)
Output: Secure encryption with key k and plaintext p if $p \neq \hat{p}$; k if $p = \hat{p}$

$n \leftarrow 4$ ▷ Number of Feistel rounds
for $i \leftarrow 0$ to $n-1$ **do**
  $K_i \leftarrow \texttt{AES}_k(i)$ ▷ Generate Feistel round keys by using `AES` in `CTR` mode
$K_4, K_5 \leftarrow k[0:63], k[64:127]$ ▷ Split k to use as post-whitening keys
$\hat{p_0}, \hat{p_1} \leftarrow \hat{p}[0:63], \hat{p}[64:127]$ ▷ Split $\hat{p}$ into 2 parts
Pre-compute and store $\texttt{SHAKE}(\hat{p_0})$, $\texttt{SHAKE}(\hat{p_1})$ ▷ Both are of 128-bits
$C_0, C_1 \leftarrow p[0:63], p[64:127]$ ▷ Split p into 2 parts
for $i \leftarrow 0$ to $n-1$ **do** ▷ Iterate over Feistel rounds
  $j \leftarrow i \pmod 2$
  $\alpha_i \leftarrow \texttt{AES}_{K_i}(\texttt{SHAKE}(\hat{p_j}))[0:63]$
  $\beta_i \leftarrow \texttt{AES}_{K_i}(\texttt{SHAKE}(C_0))[0:63]$ ▷ $\texttt{SHAKE}(C_0)$ is of 128-bits
  $f_i \leftarrow \beta_i \oplus \alpha_i$ ▷ $f_i = 0$ when $C_0 = \hat{p_j}$
  $C_1 \leftarrow C_1 \oplus f_i$ ▷ Update right Feistel branch with f_i
  if $i \leq n-2$ **then** ▷ No branch swap in last Feistel round
    $C_0, C_1 \leftarrow C_1, C_0$ ▷ Swap two Feistel branches
$C_0, C_1 \leftarrow C_0 \oplus K_4, C_1 \oplus K_5$ ▷ XOR post-whitening keys
return $C_0 || C_1$ ▷ $(C_0, C_1) = (\hat{p_1} \oplus K_4, \hat{p_0} \oplus K_5)$ when $p = \hat{p}$

well-analysed primitives—all the concepts used in its construction/analysis are known for decades. Its security can be formally proven.
2. `Malicious` requires a tweak. Tweak is said to be relatively new, less efficient, and any standard does not appear to exist [2, Section II.B]. `ZUGZWANG` does not require a tweak.
3. The key recovery using the secret backdoor entry in `ZUGZWANG` is deterministic in nature, which requires only one call to the cipher with the secret backdoor entry equated with the plaintext; whereas `LOWMC-M` requires multiple calls, and the key recovery is not guaranteed even if its queries satisfy all the requisite conditions.
4. The overall idea of `ZUGZWANG` is easier to visualise, analyse and implement. It is not clear whether the designers [9] actually have constructed the full cipher, or have left it as a wishful thinking – thus it is further not clear whether `Malicious` would work in real life.

5 Conclusion

Taking inspiration from Peyrin and Wang's Crypto'20 paper [9], we partake in a deeper dive at backdoor construction and related security concerns. A major contribution in our work is to present a block cipher concept, `ZUGZWANG`, that has an internal backdoor it. We also show a concrete instance of the concept. Our construction answers some of the open problems of `Malicious`/`LOWMC-M` [9], thus considerably improving from it.

To make ourselves clear, we do not support the government or any organisation forcing/tricking anybody to use any cipher that has a backdoor. We believe the intentional design a cipher with a hidden backdoor should be done as an academic curiosity (and not for any practical application).

References

1. Albrecht, M., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. Cryptology ePrint Archive, Report 2016/687 (2016). https://eprint.iacr.org/2016/687
2. Baksi, A., Bhasin, S., Breier, J., Khairallah, M., Peyrin, T.: Protecting block ciphers against differential fault attacks without re-keying (extended version). Cryptology ePrint Archive, Report 2018/085 (2018). https://eprint.iacr.org/2018/085
3. Baksi, A., Bhattacharjee, A., Breier, J., Isobe, T., Nandi, M.: Big brother is watching you: a closer look at backdoor construction. Cryptology ePrint Archive, Paper 2022/953 (2022). https://eprint.iacr.org/2022/953
4. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The Simon and speck families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404 (2013). https://eprint.iacr.org/2013/404
5. Chow, S., Eisen, P., Johnson, H., van Oorschot, P.C.: A white-box DES implementation for DRM applications. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 1–15. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-44993-5_1
6. Chow, S., Eisen, P., Johnson, H., Van Oorschot, P.C.: White-box cryptography and an AES implementation. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 250–270. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36492-7_17
7. Liu, F., Isobe, T., Meier, W.: Cryptanalysis of full LowMC and LowMC-M with algebraic techniques. Cryptology ePrint Archive, Paper 2020/1034 (2020). https://eprint.iacr.org/2020/1034
8. Nohl, K., Evans, D., Starbug, Plötz, H.: Reverse-engineering a cryptographic RFID tag. In: van Oorschot, P.C. (ed.) Proceedings of the 17th USENIX Security Symposium, July 28–August 1, 2008, San Jose, CA, USA, pp. 185–194. USENIX Association (2008). https://www.usenix.org/events/sec08/tech/full_papers/nohl/nohl.pdf
9. Peyrin, T., Wang, H.: The malicious framework: Embedding backdoors into tweakable block ciphers. Cryptology ePrint Archive, Report 2020/986 (2020). https://eprint.iacr.org/2020/986
10. Ravi, P., Deb, S., Baksi, A., Chattopadhyay, A., Bhasin, S., Mendelson, A.: On threat of hardware trojan to post-quantum lattice-based schemes: a key recovery attack on saber and beyond. In: Batina, L., Picek, S., Mondal, M. (eds.) SPACE 2021. LNCS, vol. 13162, pp. 81–103. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-95085-9_5
11. Shamir, A.: How to share a secret. Commun. ACM. **22**, 612–613 (1979). https://dl.acm.org/doi/10.1145/359168.359176